

ALLEGHENY COLLEGE
COMPUTER AND INFORMATION SCIENCE DEPARTMENT

Senior Thesis

Pseudo - Simulation of Omni Model Behavior

by

Alish Chhetri

ALLEGHENY COLLEGE

DEPARTMENT OF COMPUTER AND
INFORMATION SCIENCE

Project Supervisor: **Emily Graber**
Co-Supervisor: **Janyl Jumadinova**

Abstract

The artificial intelligence landscape today presents two divergent approaches: monolithic “omni-models” pursuing general capabilities through massive scaling, and specialized models that excel within specific domains. This thesis introduces Pseudo, a system that explores bridging these approaches through specialized model routing.

Pseudo investigates how specialized models might be orchestrated while maintaining a unified interface. While the theoretical framework proposes sophisticated content analysis for routing requests, the current implementation uses deepseek-r1 through Ollama to classify user inputs and determine appropriate modalities. During development, the fragmentation of AI service provider APIs emerged as a significant challenge, leading to the creation of APICenter, a universal abstraction layer addressing the broader “provider abstraction problem” faced by developers working with multiple AI services.

This research examines both the theoretical potential of intelligent routing to specialized models and the practical challenges encountered in early implementation. The findings suggest that while the current implementation has significant limitations, the approach of intelligent orchestration of specialized components could potentially offer advantages in resource efficiency, provider flexibility, specialized task performance, and ethical diversity. This represents an alternative direction to the prevailing assumption that increasingly large models are the optimal path toward advanced AI capabilities.

Table of contents

0.1	Chapter 1: Introduction	5
0.1.1	1.1 Context and Background	5
0.1.2	1.2 Problem Statement	9
0.1.3	1.3 Research Questions and Motivations	11
0.1.4	1.4 Research Contributions and Thesis Structure	20
0.2	Chapter 2: Related Work	21
0.2.1	2.1 Evolution of Foundation Models and Their Limitations	21
0.2.2	2.2 Specialized AI Services and Systems	23
0.2.3	2.3 Orchestration Approaches and API Standardization Efforts	26
0.3	Chapter 3: Methods	34
0.3.1	3.1 Architecture and Design Principles	34
0.3.2	3.2 APICenter: Provider Abstraction Implementation	40
0.3.3	3.3 Pseudo: Intelligent Routing System	42
0.3.4	3.4 Implementation Challenges and Solutions	49
0.3.5	3.5 Chapter Summary	52
0.4	Chapter 4: Experiments	52
0.4.1	4.1 Research Methodology and Experimental Design	53
0.4.2	4.2 Classification Performance and Analysis	54
0.4.3	4.3 System Limitations and Future Improvements	57
0.5	Chapter 5: Future Work	58
0.5.1	5.1 Ethical and Environmental Implications	59
0.5.2	5.2 Technical Improvements and Research Directions	61
0.5.3	5.3 Long-Term Vision and Societal Impact	63
0.5.4	5.4 Conclusion	65
0.5.5	5.5 Current Implementations	68

List of Figures

List of Tables

0.1 Chapter 1: Introduction

0.1.1 1.1 Context and Background

Artificial intelligence has reached a critical inflection point characterized by two distinct and increasingly divergent developmental paths [11], [48]. This bifurcation represents not merely a technical divergence but a philosophical one that shapes how we conceptualize the future of intelligent systems. The seminal work of Devlin et al. [11] with BERT demonstrated how powerful representations could emerge from deep bidirectional training, establishing a foundation for subsequent scaling approaches. Their research showed that pre-trained representations could dramatically improve performance across a wide range of NLP tasks without substantial task-specific architectural modifications, planting the seeds for the scaling hypothesis that would come to dominate much of AI research.

The contrast between these two developmental paths can be understood through a human-centered metaphor: the difference between a “know-it-all” generalist and a collaborative “think tank” of specialists.

The first developmental path—akin to creating a singular “know-it-all”—pursues Artificial General Intelligence (AGI, a hypothetical intelligence that can understand, learn, and apply knowledge across any domain at human or superhuman levels) through ever-larger monolithic models, exemplified by systems like GPT-4 [41], Claude Opus, and Gemini [10]. These “omni-models” handle diverse tasks across multiple domains and modalities within unified architectures, offering unprecedented versatility and convenience to users. Like a “know-it-all” individual who claims expertise in everything from quantum physics to culinary arts, these models attempt to master all domains through sheer scale and breadth of training. The architecture of these systems reflects what Sevilla et al. [41] identified as the “bigger-is-better” paradigm, where increased parameter counts correlate strongly with improved capabilities across virtually all measured dimensions. Their comprehensive analysis of compute trends across three eras of machine learning reveals an exponential increase in computational resources devoted to training these models, with a 10,000x increase in compute usage between 2010 and 2022. This trajectory suggests that the industry has committed heavily to a path where scale is the primary driver of capability improvement, with AGI as the ultimate goal, despite the substantial resource implications of this approach.

The second path—resembling a collaborative “think tank” of domain experts—creates specialized models optimized for specific domains, producing systems with exceptional performance within narrowly defined contexts but limited applicability beyond their specializations. Just as a think tank might assemble experts in economics, foreign policy, and environmental science to collaborate on complex problems, this approach leverages the strengths of purpose-built models working in concert. Models like CLIP [33] exemplify this approach; Radford et al. [33] demonstrated that a specialized architecture trained specifically on image-text pairs could achieve remarkable zero-shot performance on visual recognition tasks while requiring significantly fewer parameters than general-purpose alternatives. Their research showed that purpose-built architectures with focused training objectives could achieve superior results in specific domains, challenging the assumption that general capabilities must emerge from undifferentiated scale. Similarly, the Whisper model [34] for speech recognition exhibits how domain-specific architectural choices and training methodologies produce state-of-the-art results in audio transcription across numerous languages with relatively modest computational requirements compared to generalist counterparts.

This dichotomy, extensively documented by Bommasani et al. [5] and analyzed by

Chowdhery et al. [10], reflects a fundamental tension in AI system design that extends beyond technical considerations to encompass broader questions about resource allocation, accessibility, and the democratization of AI capabilities. Bommasani et al. [5] provided one of the most comprehensive analyses of what they termed “foundation models,” examining their opportunities and risks across technical, societal, and ethical dimensions. Their work identified how these large, adaptable models create both unprecedented capabilities and significant challenges, including issues of centralization, homogenization, and resource concentration. This analysis frames the current bifurcation not merely as a technical choice but as one with profound implications for how AI capabilities will be distributed and controlled within society.

Chowdhery et al. [10] further expanded this analysis through their development and evaluation of the PaLM model, demonstrating the remarkable emergent capabilities that arise as models scale to hundreds of billions of parameters. Their detailed benchmarking across 29 different English NLP tasks revealed that performance improvements from scale followed predictable power laws, supporting the scaling hypothesis that has driven the development of monolithic approaches. However, their work also acknowledged the practical limitations of this approach, noting that “the resource requirements for training the largest language models restrict their development to organizations with extraordinary computational capabilities,” foreshadowing questions about accessibility and concentration of power that have become increasingly central to discussions about AI development.

The trajectory toward monolithic omni-models has been driven by compelling empirical evidence that increased scale correlates with emergent capabilities [20, Wei et al. [48]]. This correlation, first systematically documented by Kaplan et al. [20] in their influential paper on scaling laws for neural language models, established mathematical relationships between model size, dataset size, and performance that have guided subsequent research and development. Their work demonstrated that performance improvements follow predictable power laws as models scale, suggesting that continued increases in parameter counts would yield proportional improvements in capabilities. This finding has had profound implications for the field, essentially creating a roadmap that many organizations have followed: invest in larger models to achieve better results.

As parameter counts have grown from millions to billions and now trillions, models demonstrate increasingly surprising abilities to handle complex tasks for which they were not explicitly trained. Wei et al. [48] provided one of the most rigorous analyses of this phenomenon in their research on emergent abilities in large language models. They identified specific capabilities—such as multi-step reasoning, following nuanced instructions, and in-context learning—that appear only after crossing certain parameter thresholds, exhibiting what they termed “emergent” behavior. Their work demonstrated that these capabilities do not gradually improve with scale but instead appear suddenly at specific thresholds, creating discontinuities in the performance curves that had not been predicted by earlier scaling laws. This observation has fueled what some researchers have termed an “arms race” toward massive architectures, with each generation of models surpassing its predecessors in both size and capabilities.

The implications of Wei et al.’s [48] findings extend beyond technical considerations to shape the strategic direction of AI research. Their documentation of emergent capabilities has reinforced the notion that scale itself is a viable path toward artificial general intelligence (AGI), leading many organizations to prioritize increased model size over architectural innovations. This perspective has been further reinforced by the remarkable capabilities of models like GPT-4 [41], which demonstrate broad competencies across domains ranging from

mathematical reasoning to creative writing to visual understanding, suggesting that sufficient scale might eventually yield systems with human-like general intelligence. The pursuit of AGI through scale alone represents a significant departure from earlier approaches to AI that focused on specialized systems for specific domains, reflecting a philosophical shift in how researchers conceptualize intelligence and its computational implementation.

However, this scaling trajectory has not progressed without significant critiques. Hoffmann et al. [16] challenged the original scaling laws proposed by Kaplan, demonstrating that optimal performance requires balancing model size with dataset size rather than simply increasing parameters. Their research suggested that many models were actually “under-trained” relative to their size, indicating inefficiencies in the prevailing approach to scaling. Similarly, Sevilla et al. [41] documented diminishing returns in model performance as scale increases, with each increment of improvement requiring exponentially greater computational resources. These critiques suggest potential natural limits to the scaling approach, raising questions about its long-term viability as the primary driver of AI advancement.

Parallel to this scaling trend, specialized models optimized for specific modalities have continued to advance along a different trajectory, often achieving superior performance within their domains while requiring substantially fewer resources. These purpose-built architectures represent an alternative philosophy to AI development—one that prioritizes targeted excellence over general capabilities, and efficiency over scale. This approach has yielded remarkable results across various domains, challenging the assumption that monolithic scaling is the only viable path to advanced AI capabilities.

The CLIP (Contrastive Language-Image Pre-training) model [33] exemplifies this specialized approach in the visual-linguistic domain. Radford et al. [33] introduced a novel training methodology that paired images with their textual descriptions, creating a system that could understand both visual and textual information through a joint embedding space. With 400 million image-text pairs and a relatively modest parameter count compared to general-purpose models, CLIP achieved remarkable zero-shot performance across a wide range of visual classification tasks. Their research demonstrated that a specialized architecture with a focused objective function could outperform much larger general-purpose models on specific tasks. The CLIP architecture’s efficiency stems from its targeted design—rather than attempting to master all possible tasks, it focuses exclusively on connecting visual and linguistic representation spaces, allowing it to excel at tasks that leverage this specific capability. This specialization enabled CLIP to achieve state-of-the-art performance in zero-shot image classification with significantly lower computational requirements than alternatives, demonstrating how architectural focus can yield both efficiency and effectiveness.

In the domain of image generation, DALL-E [38] and its successor DALL-E 2 further illustrate the power of specialized architectures. Ramesh et al. [38] developed a system specifically optimized for generating images from textual descriptions, using a unique two-stage approach that first creates a latent representation and then generates high-resolution images. Their work demonstrated that by focusing exclusively on the image generation task, they could create a system that produces remarkably detailed and accurate visualizations from text prompts. The specialized nature of DALL-E allows it to incorporate domain-specific optimizations that would be impractical in a general-purpose model, such as the use of prior distributions specifically tuned for visual content and specialized decoding strategies optimized for image generation. These architectural choices reflect a deep understanding of the specific challenges of text-to-image generation, resulting in a system that achieves superior results with greater efficiency than would be possible with a general-purpose approach.

The Whisper model [34] for automatic speech recognition further demonstrates how

domain specialization drives excellence. Radford et al. [34] created a system specifically designed for transcribing spoken language across multiple languages and acoustic environments. By training on 680,000 hours of multilingual and multitask supervised data, they produced a model that achieves robust performance across 96 languages and various acoustic conditions. The architecture incorporates specific design choices optimized for audio processing, including specialized encoder-decoder structures and training objectives tailored to speech recognition challenges. These domain-specific optimizations enabled Whisper to achieve remarkable robustness to accents, background noise, and technical language—areas where general-purpose models often struggle. The specialized nature of Whisper allowed its designers to incorporate acoustic knowledge and linguistic patterns specific to speech recognition, creating a system that outperforms general models despite more modest computational requirements.

In the programming domain, Codex [8] represents another successful application of the specialization approach. Chen et al. [8] fine-tuned a large language model specifically on code repositories, creating a system optimized for code generation and understanding. This domain-specific training enabled Codex to develop a deep understanding of programming concepts, syntax, and patterns across multiple languages. The specialized nature of Codex allows it to incorporate domain-specific knowledge about programming best practices, common algorithms, and language-specific idioms—knowledge that would be diluted in a truly general-purpose model. This specialization results in superior performance on programming tasks compared to general-purpose alternatives of similar size, demonstrating again how focused training and architecture can yield more efficient and effective systems.

These specialized models share a common philosophy that contrasts sharply with the monolithic scaling approach: they prioritize depth within specific domains over breadth across all domains. By focusing architectural choices, training data, and optimization objectives on particular tasks, these models achieve remarkable performance within their specialties while requiring significantly fewer parameters and computational resources than their monolithic counterparts. This efficiency stems from their ability to incorporate domain-specific knowledge and optimizations that would be impractical or ineffective in a truly general-purpose system.

The contrast between these approaches—monolithic scaling versus specialized excellence—represents a fundamental tension in advanced AI development that extends beyond technical considerations to encompass philosophical perspectives on the nature of intelligence itself. While monolithic models pursue artificial general intelligence through scale and homogeneity, specialized models implicitly suggest that intelligence is inherently modular and domain-specific—a perspective aligned with certain theories of human cognition that emphasize specialized neural circuits for different cognitive functions.

From a practical perspective, this tension creates a challenging tradeoff for AI developers and users. Monolithic models offer convenience and generality, providing a single interface for diverse tasks, but they sacrifice efficiency, specialized performance, and accessibility due to enormous resource requirements. Conversely, specialized models excel in performance and efficiency within their domains but fragment the user experience across multiple disconnected interfaces, creating cognitive overhead and integration challenges for users and developers. This forced choice between unified mediocrity and fragmented excellence represents a significant limitation in current AI system design.

This thesis proposes that this dichotomy presents a false choice that can be resolved through intelligent orchestration. Rather than choosing between a single “know-it-all” or an uncoordinated collection of specialists, we can create a system that functions like a

well-managed think tank with a unified interface—where specialized expertise is seamlessly coordinated behind the scenes while presenting a coherent front to the user. Through sophisticated routing and abstraction mechanisms, we can potentially combine the seamless user experience of monolithic models with the superior performance and efficiency of specialized models. By dynamically directing requests to purpose-built systems based on content analysis while maintaining a unified interface, we can create a system that offers the best of both approaches—specialized excellence without fragmented experiences. This approach fundamentally challenges whether the monolithic path toward AGI is the only viable trajectory for advanced AI capabilities. The specialized model approach provides the essential foundation for this vision, offering the performance and efficiency advantages that make the orchestration approach viable and valuable.

0.1.2 1.2 Problem Statement

Despite their impressive achievements and remarkable capabilities, monolithic “omni-models” face several fundamental limitations that cannot be overcome through further scaling alone, as comprehensively argued by Sevilla et al. [41]. These constraints are not merely temporary technological hurdles but rather inherent challenges to the monolithic approach to AI architecture—challenges that originate from the philosophical and architectural assumptions underlying these systems. Much like how a “know-it-all” individual inevitably has gaps in their knowledge despite broad claims of expertise, these monolithic systems face inherent limitations that scale alone cannot address. Understanding these limitations is essential for conceptualizing alternative approaches that might address them while preserving the advantages that make monolithic models attractive.

The most immediately apparent limitation of monolithic models is their extraordinary resource intensity in both training and inference phases. The computational requirements for these systems have grown exponentially with each generation, creating substantial barriers to development, deployment, and access. Training state-of-the-art models like GPT-4 required thousands of high-performance GPUs operating continuously for months, with enormous energy consumption and infrastructure requirements. Sevilla et al. [41] conducted a detailed analysis of compute trends across three eras of machine learning, revealing that the computational resources required for training frontier models increased by a factor of 10,000x between 2010 and 2022. Their research estimated that training GPT-4 likely required compute resources exceeding 10^{25} floating point operations, with costs conservatively estimated to exceed \$100 million. This extraordinary investment places the development of these models beyond the reach of all but the most well-resourced organizations, creating a concerning concentration of capabilities.

This computational burden extends beyond the training phase to inference as well, requiring substantial infrastructure with significant financial and environmental costs for ongoing operation. While inference costs per token are lower than training costs, the sheer scale of deployment multiplies these costs across billions of interactions. Deployment of these models at scale requires specialized hardware, extensive optimization, and substantial engineering resources, creating barriers that exclude many potential users and use cases. The resource intensity of inference also limits deployment options, making edge computing and resource-constrained environments impractical for these models, despite the significant advantages such deployments might offer for privacy, latency, and accessibility.

The environmental implications of this computational intensity are particularly concerning and have been rigorously documented by several researchers. In their groundbreaking

analysis, Strubell et al. [44] conducted one of the first comprehensive assessments of the environmental impact of large neural network training. Their research quantified the carbon footprint of developing natural language processing models, finding that training a single transformer model with neural architecture search could produce carbon emissions equivalent to the lifetime emissions of five average American cars. They further demonstrated that these emissions grow proportionally with model size, dataset size, and hyperparameter optimization efforts, suggesting that the environmental costs of the scaling approach would continue to increase at an alarming rate. Their work raised essential questions about the environmental sustainability of the prevailing scaling paradigm in AI research, questions that have become increasingly urgent as models have continued to grow.

Building on this foundation, Patterson et al. [32] provided a more detailed analysis of the carbon emissions associated with large neural network training, with a specific focus on language models. Their research estimated that training a GPT-3-scale model consumes approximately 1,287 MWh of electricity, equivalent to the annual energy consumption of 284 average American homes. This analysis incorporated not only the direct energy consumption of the training hardware but also the energy required for cooling, infrastructure, and related operations, providing a more comprehensive view of the environmental impact. Patterson’s work highlighted the substantial gap between the most and least efficient training approaches, demonstrating that architectural and operational choices can significantly influence environmental impact. Their research emphasized the importance of carbon-aware computing practices, suggesting that strategic choices about when, where, and how models are trained could substantially reduce environmental impacts without compromising capabilities.

Luccioni et al. [32] further extended this analysis with a more recent and detailed quantification of the environmental costs of foundation model training. Their comprehensive methodology accounted for variations in hardware efficiency, data center Power Usage Effectiveness (PUE), and regional electricity carbon intensity, producing more accurate estimates of carbon emissions. They demonstrated that training a single state-of-the-art foundation model can generate between 25 and 500 metric tons of CO₂ equivalent emissions, depending on the specific hardware, location, and model architecture. Their research highlighted the significant differences in carbon intensity between different regions, suggesting that strategic decisions about training locations could reduce emissions by a factor of 30 or more without changing the model itself. This geographic variability creates both challenges and opportunities for environmentally conscious AI development, emphasizing that deployment decisions can be as important as architectural choices in determining environmental impact.

Collectively, these analyses paint a concerning picture of the resource requirements and environmental impacts of the scaling approach to AI development. As models continue to increase in size and complexity, these impacts grow proportionally, raising serious questions about the long-term sustainability of this trajectory. The environmental costs of AI development represent not merely an academic concern but a concrete limitation that must be addressed if AI is to fulfill its potential as a broadly beneficial technology. These findings suggest an urgent need for alternative approaches that can deliver advanced AI capabilities with substantially lower resource requirements and environmental impacts.

Dependency on single providers creates problematic lock-in effects for users and developers, as documented by Birhane et al. [4]. Organizations building critical systems on these models become vulnerable to changes in provider policies, pricing, or availability. This dependency raises concerns about system resilience and sustainability, particularly for applications requiring long-term stability or operation in diverse regulatory contexts. Drexler

[3] explored the economic implications of this centralization, arguing that concentration of AI capabilities in few providers creates significant market inefficiencies and innovation barriers.

Collectively, these analyses paint a concerning picture of the resource requirements and environmental impacts of the scaling approach to AI development. As models continue to increase in size and complexity, these impacts grow proportionally, raising serious questions about the long-term sustainability of this trajectory. The environmental costs of AI development represent not merely an academic concern but a concrete limitation that must be addressed if AI is to fulfill its potential as a broadly beneficial technology. These findings suggest an urgent need for alternative approaches that can deliver advanced AI capabilities with substantially lower resource requirements and environmental impacts.

Beyond the environmental concerns, the consolidation of frontier AI development within a handful of well-resourced organizations creates significant structural dependencies that have far-reaching implications. Dependency on single providers creates problematic lock-in effects for users and developers, as comprehensively documented by Birhane et al. [4] in their analysis of power dynamics in machine learning research. Their work examines how architectural decisions and deployment patterns in AI create asymmetric relationships between providers and users, with providers maintaining disproportionate control over system behavior, availability, and evolution. Organizations building critical systems atop these models become increasingly vulnerable to unilateral changes in provider policies, pricing structures, or service availability—changes over which they have little influence despite potentially significant downstream impacts on their operations.

This dependency raises substantial concerns about system resilience and sustainability, particularly for applications requiring long-term stability or operation in diverse regulatory contexts where provider priorities may conflict with local requirements. The technical complexity and proprietary nature of these systems make transitions between providers extremely costly and disruptive, creating what economists term “high switching costs” that further entrench dependency relationships. These switching costs stem from various sources, including API incompatibilities, differences in model behavior and capabilities, accumulated organizational expertise specific to particular providers, and the substantial engineering resources required to migrate between systems.

Drexler [3] explored the economic implications of this centralization in depth, applying concepts from industrial organization theory to identify concerning patterns in AI development ecosystems. His analysis demonstrates how concentration of AI capabilities within few providers creates significant market inefficiencies that potentially impede broader innovation. This concentration reduces diversity of approaches, constrains experimentation with alternative architectures, and potentially directs research attention toward incremental improvements in established paradigms rather than transformative new approaches. The resulting innovation bottlenecks could potentially slow overall progress in the field, particularly for applications or approaches that don’t align with the priorities of dominant providers. This market concentration also raises concerns about pricing power, potentially enabling providers to extract disproportionate value relative to their contributions and limiting broader access to advanced capabilities.

0.1.3 1.3 Research Questions and Motivations

Collectively, these analyses paint a concerning picture of the resource requirements and environmental impacts of the scaling approach to AI development. As models continue to

increase in size and complexity, these impacts grow proportionally, raising serious questions about the long-term sustainability of this trajectory. The environmental costs of AI development represent not merely an academic concern but a concrete limitation that must be addressed if AI is to fulfill its potential as a broadly beneficial technology. These findings suggest an urgent need for alternative approaches that can deliver advanced AI capabilities with substantially lower resource requirements and environmental impacts.

Beyond the environmental concerns, the consolidation of frontier AI development within a handful of well-resourced organizations creates significant structural dependencies that have far-reaching implications. Dependency on single providers creates problematic lock-in effects for users and developers, as comprehensively documented by Birhane et al. [4] in their analysis of power dynamics in machine learning research. Their work examines how architectural decisions and deployment patterns in AI create asymmetric relationships between providers and users, with providers maintaining disproportionate control over system behavior, availability, and evolution. Organizations building critical systems atop these models become increasingly vulnerable to unilateral changes in provider policies, pricing structures, or service availability—changes over which they have little influence despite potentially significant downstream impacts on their operations.

This dependency raises substantial concerns about system resilience and sustainability, particularly for applications requiring long-term stability or operation in diverse regulatory contexts where provider priorities may conflict with local requirements. The technical complexity and proprietary nature of these systems make transitions between providers extremely costly and disruptive, creating what economists term “high switching costs” that further entrench dependency relationships. These switching costs stem from various sources, including API incompatibilities, differences in model behavior and capabilities, accumulated organizational expertise specific to particular providers, and the substantial engineering resources required to migrate between systems.

Drexler [3] explored the economic implications of this centralization in depth, applying concepts from industrial organization theory to identify concerning patterns in AI development ecosystems. His analysis demonstrates how concentration of AI capabilities within few providers creates significant market inefficiencies that potentially impede broader innovation. This concentration reduces diversity of approaches, constrains experimentation with alternative architectures, and potentially directs research attention toward incremental improvements in established paradigms rather than transformative new approaches. The resulting innovation bottlenecks could potentially slow overall progress in the field, particularly for applications or approaches that don’t align with the priorities of dominant providers. This market concentration also raises concerns about pricing power, potentially enabling providers to extract disproportionate value relative to their contributions and limiting broader access to advanced capabilities.

Beyond provider dependency, the ethical homogeneity inherent in monolithic approaches presents equally significant challenges for diverse use cases and cultural contexts. These large, centralized models necessarily embed singular ethical frameworks and content policies that cannot adequately accommodate the rich diversity of normative perspectives that exist across different communities, domains, and regions. Weidinger et al. [49] provide one of the most comprehensive analyses of this ethical homogeneity problem in their exploration of language model risks. Their research identifies how seemingly technical decisions about content filtering, bias mitigation, and prompt processing implicitly encode specific ethical judgments about what constitutes harmful, offensive, or inappropriate content—judgments that may vary dramatically across different cultural contexts.

For example, Weidinger et al. [49] document how content policies around discussions of sexuality, political systems, or religious criticism reflect primarily Western liberal perspectives that may conflict with the norms and legal requirements of other jurisdictions. This mismatch creates significant challenges for global deployment, potentially forcing organizations to choose between violating local norms and regulations or forgoing advanced AI capabilities entirely. The researchers particularly emphasize how these ethical frameworks are typically determined by the small group of engineers and researchers developing these systems, rather than through inclusive processes that incorporate diverse perspectives from affected communities. This centralization of ethical decision-making raises profound questions about representation, legitimacy, and cultural imperialism in AI development.

Bender et al. [3] further highlight how this ethical homogeneity can amplify existing biases and power imbalances, disproportionately affecting marginalized communities. Their influential “Stochastic Parrots” paper examines how large language models trained on internet data inevitably reproduce and amplify existing patterns of bias, exclusion, and misrepresentation present in that data. The authors demonstrate how these models typically overrepresent dominant perspectives while systematically underrepresenting or misrepresenting marginalized voices. This representation disparity becomes particularly problematic as these systems increasingly mediate access to information, opportunities, and resources, potentially reinforcing existing inequities rather than mitigating them.

The specialized routing approach proposed in this thesis addresses these ethical and informational homogeneity concerns by enabling dynamic routing to different models from different providers based on context-specific requirements. This architecture supports greater ethical pluralism by allowing organizations to implement context-specific routing rules that direct different types of content to providers with appropriate ethical frameworks. This architectural pluralism enables more nuanced implementation of ethical guidelines that respect the legitimate diversity of values across different communities and contexts.

One particularly powerful example of the effectiveness of specialized models can be found in CLIP (Contrastive Language-Image Pre-training) [33], which represents a landmark achievement in the development of specialized vision-language models. Radford et al. [33] introduced CLIP as a response to the limitations of traditional image classification approaches, which typically required extensive labeled datasets and struggled with zero-shot generalization to new categories. Rather than pursuing a general-purpose model with ever-increasing parameters, the researchers focused on developing a specialized architecture optimized specifically for connecting visual and linguistic representation spaces.

CLIP’s specialized architecture employs a dual-encoder approach that trains image and text encoders to maximize the similarity of correct image-text pairs while minimizing the similarity of incorrect pairings. This contrastive learning objective enables the model to learn a joint embedding space where semantically related images and texts are positioned closely together. By training on 400 million image-text pairs collected from the internet, the researchers created a system that could understand both visual and textual information through this shared representation space, without requiring manually labeled datasets or extensive task-specific fine-tuning.

The specialized nature of CLIP is evident in several key design decisions that differentiate it from general-purpose approaches. First, the architecture was specifically optimized for the cross-modal alignment task, with careful attention to the balance between image and text encoders. Rather than attempting to create a single model that could perform all possible tasks, the researchers focused exclusively on creating effective representations that connect visual and linguistic information. This focused approach allowed them to make architectural

choices specifically tailored to this cross-modal challenge, including specialized attention mechanisms and representation normalization techniques that would be suboptimal for other tasks.

Second, the training methodology was specifically designed for the vision-language alignment task, using a contrastive approach that leverages natural language supervision rather than traditional classification labels. This approach directly targets the specific challenge of connecting linguistic and visual representations, rather than attempting to solve multiple problems simultaneously. The contrastive objective function creates a more direct learning signal for the cross-modal alignment task than would be possible in a multi-task learning scenario, enabling more efficient learning from the available data.

Third, the evaluation methodology specifically focused on zero-shot transfer to downstream visual recognition tasks, rather than attempting to optimize for a broad range of capabilities simultaneously. This specialized focus allowed the researchers to make design decisions specifically optimized for transfer learning performance, potentially sacrificing capabilities in other domains to achieve superior performance in the target domain. This willingness to make domain-specific tradeoffs exemplifies the specialized approach that stands in contrast to the monolithic scaling paradigm.

The results of this specialized approach were remarkable. CLIP achieved state-of-the-art zero-shot performance across a wide range of visual classification tasks, demonstrating the power of specialized architectural design and focused training objectives. With significantly fewer parameters than contemporary general-purpose models, CLIP achieved superior performance on specific visual understanding tasks, demonstrating how architectural focus can yield both efficiency and effectiveness. The model’s ability to connect text and images enabled powerful new capabilities like zero-shot classification and open-ended image retrieval, all without requiring domain-specific fine-tuning or labeled datasets.

CLIP’s success illustrates a key thesis of this research: that specialized models designed with focused objectives and domain-specific optimizations can achieve superior performance within their domains while requiring significantly fewer resources than general-purpose alternatives. This pattern of specialized excellence has been repeated across numerous domains, from DALL-E [38] in image generation to Whisper [34] in speech recognition, suggesting that specialization represents a viable alternative path to advanced AI capabilities that complements the scaling approach.

The challenge, however, lies in integrating these specialized capabilities into unified interfaces that maintain the convenience of monolithic models while leveraging the superior performance of specialized alternatives. The routing approach proposed in this thesis directly addresses this challenge by providing intelligent orchestration of specialized models behind a unified interface, potentially combining the best aspects of both approaches. Models like CLIP could serve as specialized components within such a system, handling vision-language tasks with superior performance and efficiency while being seamlessly integrated with other specialized components for different modalities and tasks.

This thesis addresses one fundamental research question that guides the investigation into alternative approaches to AI model architecture and orchestration:

Can a routing-based approach effectively mimic the behavior of omni-models while leveraging specialized services?

This primary question explores whether it’s possible to create a system that presents users with a unified interface similar to monolithic omni-models while directing requests to specialized models based on content analysis. The investigation examines both the theoretical potential of intelligent routing and the practical challenges encountered in implementing

such a system.

The question addresses multiple dimensions of the routing approach:

1. **Classification Accuracy:** Can natural language requests be reliably classified to determine the appropriate processing modality (text, image, audio)?
2. **User Experience Continuity:** Can a routing-based system maintain the seamless experience associated with monolithic models while leveraging specialized components?
3. **Implementation Feasibility:** What practical challenges arise when implementing routing-based systems, and how might these challenges be addressed?

By exploring this fundamental question, the research aims to determine whether intelligent routing to specialized models represents a viable alternative to the prevailing paradigm of increasingly large monolithic models. This exploration has significant implications for the future development of AI systems, touching on resource efficiency, provider flexibility, specialized task performance, and potential enhancements in ethical diversity and information pluralism.

This thesis addresses one fundamental research question that guides the investigation into alternative approaches to AI model architecture and orchestration:

Can a routing-based approach effectively mimic the behavior of omni-models while leveraging specialized services?

This primary question explores whether it’s possible to create a system that presents users with a unified interface similar to monolithic omni-models while directing requests to specialized models based on content analysis. The investigation examines both the theoretical potential of intelligent routing and the practical challenges encountered in implementing such a system. The motivation behind this question arises from the recognition of the fundamental tensions between monolithic and specialized approaches documented in the literature, and the possibility that intelligent orchestration might enable a third path that captures the advantages of both while mitigating their respective limitations.

This research question builds directly upon critical observations in recent literature about the tradeoffs inherent in current AI architectures. Bommasani et al. [5] identified the critical tension between general-purpose foundation models and specialized approaches in their comprehensive analysis, noting that “foundation models provide general capabilities that can be adapted to a wide range of downstream tasks, but the generality of these capabilities may limit their effectiveness on specific tasks compared to more specialized approaches.” This observation frames the fundamental tradeoff that motivates our research: general models offer convenience but potentially sacrifice performance, while specialized models offer excellence but fragment the user experience. The possibility of resolving this tension through intelligent orchestration represents a novel contribution to this ongoing conversation in AI architecture.

Similarly, Chen et al. [8] demonstrated empirically that specialized models for code generation significantly outperform general-purpose alternatives on programming tasks despite having fewer parameters and requiring less computational resources. Their research on Codex showed that targeted architectural choices and domain-specific training yield superior results for specific tasks, suggesting that specialization offers advantages that scale alone cannot provide. This empirical evidence challenges the prevailing assumption that increasingly large general-purpose models represent the optimal approach for all tasks, creating

space for alternative architectural approaches that might better leverage these specialized capabilities while maintaining user convenience.

The research question addresses multiple dimensions of the routing approach, each motivated by specific limitations in current systems and possibilities identified in the literature:

1. **Classification Accuracy:** Can natural language requests be reliably classified to determine the appropriate processing modality (text, image, audio)? This dimension acknowledges the critical role of content understanding in effective routing, building on research by Wolf et al. [51] demonstrating that transformer-based architectures can achieve high accuracy in text classification tasks with relatively modest training data. Their work with the Hugging Face Transformers library established that even relatively small models can achieve sophisticated text understanding when trained on well-defined classification tasks, suggesting the potential feasibility of accurate modality detection without requiring enormous computational resources.
2. **User Experience Continuity:** Can a routing-based system maintain the seamless experience associated with monolithic models while leveraging specialized components? This dimension addresses a critical practical concern identified by Weidinger et al. [49] in their analysis of language model risks—that fragmented interfaces create significant usability challenges that limit adoption despite potential technical advantages. Their work emphasized that user experience considerations are not merely practical concerns but essential components of responsible AI development, particularly for systems intended for broad accessibility. The question of whether routing approaches can maintain experience continuity while leveraging specialized components thus connects directly to broader questions about accessibility and usability in AI systems.
3. **Implementation Feasibility:** What practical challenges arise when implementing routing-based systems, and how might these challenges be addressed? This dimension acknowledges the gap between theoretical potential and practical implementation, recognizing that novel architectural approaches often encounter unexpected challenges during development. The recent work by Izacard et al. [19] on retrieval-augmented language models demonstrated how seemingly straightforward architectural innovations can reveal complex implementation challenges that require creative solutions. Their research on Atlas showed that integrating specialized components (in their case, retrieval systems) with language models created novel technical challenges around context management, latency control, and interface design that were not apparent from purely theoretical analysis. This research dimension explicitly acknowledges these potential implementation gaps and incorporates them into the investigation rather than treating them as peripheral concerns.
4. **Resource Efficiency:** Can intelligent routing to specialized models achieve comparable capabilities to monolithic approaches with lower resource requirements? This dimension addresses the environmental and accessibility concerns raised by Strubell et al. [44] and Patterson et al. [32] regarding the substantial resource requirements of large model training and inference. Their research documented the significant environmental impacts of current scaling approaches, creating motivation to explore architectures that might provide similar capabilities with reduced computational footprints. This dimension connects the technical question of routing feasibility to broader concerns about AI sustainability and accessibility.

5. **Ethical Pluralism:** Can a routing architecture enable more diverse ethical implementations by directing requests to providers with different content policies and value systems? This dimension builds on critiques by Bender et al. [3] and Weidinger et al. [49] regarding the ethical homogeneity of monolithic approaches. Their work identified how single content policies struggle to accommodate the legitimate diversity of ethical perspectives across different contexts, creating motivation to explore architectures that might enable more contextually appropriate ethical implementations while maintaining unified interfaces.

The exploration of these dimensions aims to determine whether intelligent routing to specialized models represents a viable alternative to the prevailing paradigm of increasingly large monolithic models. This exploration has significant implications for the future development of AI systems, touching on resource efficiency, provider flexibility, specialized task performance, and potential enhancements in ethical diversity and information pluralism. By challenging the assumption that increased scale represents the only path to comprehensive capabilities, this research contributes to the ongoing conversation about sustainable, accessible, and effective approaches to advanced AI development.

The development of Pseudo originated from a fundamental observation: the AI landscape has become polarized between monolithic “omni-models” pursuing generality through scale and specialized models achieving domain excellence through focus. Each approach offers distinct advantages but also significant limitations. This observation sparked the conceptualization of a system that could preserve the specialized excellence of purpose-built models while providing users with a unified interface similar to that of an omni-model—effectively simulating comprehensive capabilities through intelligent routing.

The initial vision for Pseudo was ambitious: a system that could analyze natural language requests, determine the appropriate modality and specific requirements, and direct the request to the optimal specialized model for processing. This “anything in, anything out” approach would maintain the convenience of interacting with a single system while leveraging the superior capabilities of purpose-built models for specific tasks. Early prototypes confirmed the potential of this approach but revealed a significant obstacle: the fragmentation and inconsistency of AI service provider APIs.

The practical challenges of implementing Pseudo’s vision proved more complex than initially anticipated, with provider integration representing a particularly formidable obstacle. Each major AI service provider—OpenAI, Anthropic, Stability AI, Cohere, and others—has implemented fundamentally different interfaces for their services, creating a fragmented landscape that complicates multi-provider integration. These differences extend far beyond superficial variations, encompassing fundamental aspects of service interaction that affect every level of the integration stack.

Authentication mechanisms vary dramatically across providers, from API keys passed as headers (OpenAI) to bearer tokens (Anthropic) to more complex OAuth flows required by certain services. Each authentication approach requires provider-specific implementation code, connection management, and credential handling. Some providers implement rate limiting at the authentication level, further complicating the integration by requiring specialized handling of authentication-related errors and retry logic.

Parameter naming conventions present another significant challenge, with no standardization across providers even for functionally identical parameters. A simple example illustrates this problem: the primary text input is variously called “prompt” (OpenAI), “prompt.text” (Anthropic), “text” (Cohere), or “inputs” (Hugging Face), depending on

the provider. More complex parameters show even greater divergence—temperature controls for randomness might be called “temperature,” “temperature_sampling,” or hidden within nested objects with different semantic meanings across providers. These naming inconsistencies require extensive mapping logic to translate between a consistent internal representation and provider-specific parameter formats.

Response formats show perhaps the greatest variation, with fundamentally different structures for returning essentially identical information. Some providers return flat JSON structures, others deeply nested objects with provider-specific terminology. Streaming implementations vary from server-sent events to custom WebSocket protocols to chunked HTTP responses, each requiring different client-side handling. Error responses follow no consistent pattern across providers, with different error codes, message formats, and retry semantics. Some providers include detailed diagnostic information in errors while others return minimal details, requiring different approaches to error handling and recovery.

Even seemingly straightforward aspects like URL construction and endpoint selection vary significantly across providers. Some implement clean REST-like interfaces with logical resource paths, while others use a single endpoint with complex body parameters to distinguish between services. These infrastructure differences require provider-specific client code for service discovery, endpoint selection, and request construction.

The practical impact of this fragmentation became immediately apparent in early Pseudo prototypes. Each new provider integration required substantial custom code, effectively creating independent client implementations with minimal shared logic. This provider-specific code quickly threatened to overwhelm the core routing logic of Pseudo, undermining its architectural clarity and maintainability. More concerning, the development overhead for each new provider integration created a practical limit on how many specialized services could reasonably be incorporated, potentially constraining the system’s access to specialized excellence.

As the integration complexity increased, a clear pattern emerged: the challenge of provider abstraction represented a distinct and separable problem from the challenge of intelligent routing. The code required to normalize interactions across different providers followed consistent patterns that had little to do with the content analysis and routing logic core to Pseudo’s purpose. This recognition led to the architectural decision to separate these concerns into distinct components—a decision that would ultimately shape the development of both Pseudo and APICenter.

This challenge necessitated the creation of APICenter, a universal Python interface designed to standardize interactions with diverse AI services. Initially conceived as a supportive component for Pseudo, APICenter was developed to provide consistent access patterns for text, image, and audio generation across multiple providers. The core design principle was simplicity and consistency: developers should be able to use identical patterns regardless of the underlying provider, with APICenter handling the complexities of different implementations behind a clean abstraction layer.

APICenter’s design directly addressed the practical challenges encountered during Pseudo development. The authentication diversity problem was solved through a unified credential management system that handles provider-specific authentication approaches behind a consistent interface. Developers specify credentials in a standardized format, and APICenter handles the transformation to provider-specific authentication flows transparently. This approach isolates authentication complexity from both application code and routing logic, enabling clean separation of concerns.

The parameter naming challenge was addressed through a comprehensive parameter

transformation system. APICenter defines a canonical parameter set for each modality (text, image, audio) and implements bidirectional mapping between these canonical parameters and provider-specific formats. This transformation happens transparently, allowing application code to use consistent parameter names regardless of the underlying provider. The transformation system also handles parameter constraints and validation, ensuring that provider-specific requirements are met without exposing these details to application code.

Response format inconsistencies are managed through a response standardization layer that transforms diverse provider formats into consistent structures organized by modality rather than provider. This standardization preserves all relevant information while presenting it in a provider-agnostic format that simplifies application code. Error handling receives similar treatment, with provider-specific errors mapped to a consistent exception hierarchy that maintains error details while providing uniform handling patterns.

The endpoint and infrastructure variations are encapsulated within provider-specific adapter implementations that conform to consistent interfaces defined by APICenter’s architecture. These adapters handle the details of endpoint selection, request construction, and response parsing, while exposing only the standardized interfaces to application code. This adapter pattern follows established software engineering principles, particularly the Interface Segregation Principle identified by Martin [27], which advocates that “clients should not be forced to depend upon interfaces that they do not use.”

By systematically addressing each dimension of provider inconsistency, APICenter creates a unified abstraction layer that isolates application code from provider-specific implementation details. This abstraction enables Pseudo to focus exclusively on its core intelligent routing capabilities without being burdened by the complexities of provider integration. The separation of concerns creates a cleaner architecture that enhances maintainability, testability, and extensibility for both components.

While tools like LiteLLM had already made progress in standardizing access to text-based language models using OpenAI-compatible interfaces, they did not fully address the multimodal scope or provider-agnostic architecture needed for Pseudo’s routing system. APICenter extended beyond text generation to create a truly comprehensive abstraction that could support the full range of modalities required for an “anything in, anything out” interface. This broader scope distinguished APICenter from existing solutions while building upon the proven value of standardized provider interfaces.

As development progressed, it became apparent that APICenter had value beyond its supporting role for Pseudo. The provider abstraction problem it addressed was widespread in the AI development ecosystem, affecting many developers attempting to work with multiple models or providers. This recognition led to a pivotal decision: to develop APICenter as a standalone project with its own identity, documentation, and community focus, while maintaining its integration with Pseudo.

The relationship between Pseudo and APICenter thus evolved into a complementary pair of projects with distinct but interconnected purposes. APICenter solves the “provider abstraction problem” by providing a consistent interface for accessing various AI capabilities regardless of the underlying provider, allowing developers to write code once and deploy it across multiple AI services without modification. Pseudo builds upon this foundation to implement the intelligence needed to route requests to the most appropriate specialized models based on content analysis.

This evolution reflects a common pattern in software development, where solving a specific problem leads to the identification of a more general challenge that warrants dedicated attention. By separating provider abstraction (APICenter) from intelligent routing

(Pseudo), the resulting architecture achieves greater modularity, reusability, and conceptual clarity than would have been possible with a monolithic approach.

0.1.4 1.4 Research Contributions and Thesis Structure

This research makes several significant contributions to the field of artificial intelligence and systems design:

First, the research introduces a theoretical framework for content classification and routing based on natural language inputs. While the current implementation uses a straightforward LLM-based approach, the theoretical framework proposes more sophisticated techniques that could combine pattern-based classification, keyword analysis, semantic similarity, and machine learning to accurately identify the intent and requirements of user requests. This framework provides a foundation for future development of more advanced classification systems.

Second, the development of APICenter addresses the “provider abstraction problem” by providing a standardized interface for accessing various AI services. By implementing consistent patterns for text, image, and audio generation across multiple providers, APICenter reduces the complexity and overhead associated with multi-provider integration. The design principles of simplicity, consistency, and provider-agnosticism provide a valuable model for addressing API fragmentation in the broader AI ecosystem.

Third, the current implementation of Pseudo demonstrates the practical feasibility of a routing-based approach to AI orchestration. By using a simple but effective content classification system based on deepseek-r1 through Ollama, Pseudo shows how natural language inputs can be directed to appropriate specialized models based on their content characteristics. This implementation, while limited in sophistication, provides a proof of concept for the theoretical approach.

Fourth, this research explores the potential advantages of a routing-based approach beyond the technical feasibility. By enabling access to multiple specialized models and providers through a unified interface, the approach could contribute to greater ethical diversity, reduced provider lock-in, enhanced information pluralism, and improved resource efficiency. These potential benefits suggest that routing-based approaches could address some of the social and ethical concerns associated with the concentration of AI capabilities in monolithic models.

Finally, this research contributes to the broader conversation about the future evolution of AI by proposing a third path between the dominant paradigms of monolithic scaling and specialized fragmentation. By demonstrating a viable approach that combines the advantages of both paradigms while mitigating their respective limitations, this work suggests an alternative direction for AI development that could reshape how we conceptualize and implement advanced AI systems.

The remainder of this thesis is organized to provide a comprehensive examination of Pseudo and APICenter, their design principles, implementation details, and empirical evaluation:

Chapter 2 (Related Work) situates the current research within the broader landscape of AI development, examining the evolution of foundation models, the achievements of specialized models in various domains, existing orchestration approaches, and current API abstraction efforts. This chapter identifies the research gaps that motivated the development of Pseudo and APICenter and establishes the theoretical foundation for the work presented in subsequent chapters.

Chapter 3 (Methods) details the system architecture of both Pseudo and APICenter, including their design principles, implementation details, and the technical challenges encountered during development. The chapter examines the classification system that forms the core of Pseudo’s routing capabilities, the provider selection algorithm that determines optimal routing paths, and the unified API abstraction layer implemented in APICenter. By providing a thorough exposition of the technical approach, this chapter establishes the methodological foundation for the empirical evaluation that follows.

Chapter 4 (Experiments) presents an empirical evaluation of the current implementation of Pseudo’s classification system, examining its accuracy in determining appropriate modalities from natural language inputs. Through controlled testing with diverse input types, these experiments provide evidence regarding the feasibility of the routing approach and identify areas for future improvement.

Chapter 5 (Future Work) explores the broader implications of this research for AI development, including ethical considerations, provider diversity, information pluralism, and potential directions for future work. The chapter examines both short-term improvements to the current implementation and longer-term research directions that build upon the foundation established by Pseudo and APICenter. By connecting the specific contributions of this work to broader trends in AI development, this chapter situates the research within ongoing conversations about the future evolution of artificial intelligence.

Throughout these chapters, the thesis advances a central argument: that intelligent routing to specialized models can effectively bridge the gap between monolithic omni-models and fragmented specialized tools, offering a third path that combines the advantages of both approaches while mitigating their respective limitations. By challenging the prevailing dichotomy between scale-driven generality and domain-specific excellence, this research contributes to the development of more sustainable, flexible, and effective AI systems that preserve the advantages of specialized excellence while maintaining the convenience of a unified interface.

0.2 Chapter 2: Related Work

The development of Pseudo and APICenter is situated within a rich landscape of AI research and practice, building upon advances in foundation models, specialized architectures, orchestration frameworks, and API standardization efforts. This chapter provides a comprehensive review of relevant prior work, establishing the theoretical and technical context for the current research while identifying the gaps that motivated this investigation.

0.2.1 2.1 Evolution of Foundation Models and Their Limitations

The emergence of foundation models represents a paradigm shift in artificial intelligence. These large-scale neural network models, pre-trained on vast corpora of data, have demonstrated remarkable capabilities across diverse tasks. This section traces the development of foundation models, their key characteristics, and the inherent limitations that motivate the alternative approach proposed in this research.

0.2.1.1 Development History Modern foundation models evolved from the transformer architecture introduced by Vaswani et al. [47]. This architecture, which relies entirely on attention mechanisms rather than recurrence, enabled more efficient parallel training and better capture of long-range dependencies in sequence data. The release of

BERT [11] by Google dramatically advanced the state of natural language processing by introducing bidirectional representations and demonstrating the effectiveness of pre-training followed by fine-tuning.

The evolution of foundation models followed a progression that began even before transformers. Earlier work by Radford et al. [35] on Generative Pre-Training (GPT-1) established the paradigm of unsupervised pre-training followed by supervised fine-tuning, demonstrating substantial improvements across diverse language understanding tasks. This approach was further expanded with GPT-2 [36], which showed that larger models pre-trained on more diverse datasets could perform tasks in a zero-shot setting without explicit fine-tuning. These early models established the foundation for subsequent scaling efforts, reflecting the field’s increasing focus on larger model sizes and broader pre-training datasets.

The scaling paradigm established by GPT-3 has continued with subsequent models such as PaLM [10], Gopher [37], Chinchilla [16], and GPT-4 [41]. These models have demonstrated that increased scale correlates with improved performance across a wide range of benchmarks. As Wei et al. [48] documented, certain “emergent abilities”—capabilities not present in smaller models—appear only after crossing specific parameter thresholds.

These advancements led to increasingly large models culminating in breakthroughs. However, as Sevilla et al. [41] and Hoffmann et al. [16] have shown, scaling laws suggest diminishing returns, with each increment in performance requiring exponentially greater computational resources.

While commercial organizations initially dominated the development of the largest foundation models, the release of open-source alternatives has begun to democratize access to these technologies. Models like LLaMA from Meta AI [46], Falcon [22], and Mistral [22] have made powerful foundation models more accessible to researchers and developers.

The release of LLaMA’s weights to researchers catalyzed an ecosystem of derived models through fine-tuning and adaptation. Projects such as Alpaca [22], Vicuna [22], and Koala [22] demonstrated that relatively modest computational resources could transform foundation models into specialized assistants with capabilities approaching those of commercial systems. This proliferation of open models has dramatically lowered barriers to entry for foundation model research and application development.

Modern foundation models demonstrate impressive emergent abilities, including zero- and few-shot learning abilities [7], enabling them to perform tasks with minimal task-specific examples. They show surprising transfer learning capabilities, where training on one task improves performance on seemingly unrelated tasks [48].

Perhaps most importantly, these models respond effectively to natural language instructions [22], making them accessible to users without programming expertise. By encoding complex behaviors in text prompts, users can shape model behavior without modifying model parameters, a paradigm Kojima et al. [22] termed “zero-shot reasoning.”

Recent advances in alignment techniques have further enhanced these models’ usefulness, with approaches such as Reinforcement Learning from Human Feedback (RLHF) [30], Constitutional AI [2], and Direct Preference Optimization [22]. These approaches align model outputs with human preferences and values, making the systems more helpful, harmless, and honest.

0.2.1.2 Limitations Despite their impressive capabilities, current foundation models still face significant limitations. They can produce factually incorrect information, sometimes referred to as “hallucinations” [26]. This reliability challenge becomes particularly problematic in high-stakes domains like healthcare, legal, and financial applications, where

factual accuracy is paramount. Liu et al. [26] surveyed approaches to large language model attribution, finding that even state-of-the-art models struggle to provide reliable source attribution for their generated content, complicating verification of their outputs.

Foundation models struggle with complex reasoning tasks and mathematical problem-solving [15]. As demonstrated by Hanratty et al. [15] in their assessment of LLMs for vulnerability detection, these models show promise for augmenting specialized analysis but fall short of replacing tailored, domain-specific approaches in areas requiring rigorous logical reasoning. The models remain sensitive to prompt phrasing and context structure, with small changes sometimes producing dramatically different outputs [18]. Ingebreetsen et al. [18] analyzed decoding techniques for LLMs from a theoretical perspective, revealing fundamental limitations in how these models generate text that cannot be overcome through scaling alone.

Furthermore, foundation models suffer from what Dohare et al. [22] term “loss of plasticity” - a phenomenon where neural networks become increasingly unable to learn new skills when trained sequentially on different tasks. This catastrophic forgetting presents significant challenges for developing models that can continuously adapt to new information or use cases without requiring complete retraining. The ability to maintain stability (remembering past knowledge) while preserving plasticity (learning new concepts) remains a fundamental challenge in neural network design, particularly relevant for AI systems deployed in dynamic, changing environments.

A comprehensive survey by Elkaduwe et al. [12] on decoding large language models revealed fundamental challenges in controlling their generation process to ensure reliability, factuality, and coherence. These limitations represent intrinsic challenges to the monolithic approach rather than temporary obstacles that scaling can overcome.

The high computational requirements of these models [41, Strubell et al. [44], Patterson et al. [32]] make them inaccessible to many researchers and organizations, particularly those in regions with limited computational resources. This creates an uneven playing field that concentrates AI capabilities within well-resourced institutions, primarily in North America and Western Europe.

0.2.2 Specialized AI Services and Systems

While foundation models have demonstrated remarkable general capabilities, specialized AI services designed for specific modalities or tasks often achieve superior performance, efficiency, and reliability within their domains. This section examines the landscape of specialized AI services and systems, their respective advantages, and the opportunities they present for disaggregated approaches to AI architecture.

0.2.2.1 Text Processing and Natural Language Understanding Domain-specific text processing services offer advantages even in an era dominated by general-purpose language models. Services like Named Entity Recognition (NER) [10], text classification [37], and sentiment analysis [16] can provide more reliable, deterministic, and interpretable results for well-defined tasks.

Specialized models often require significantly fewer computational resources than foundation models while delivering comparable or superior performance within their domains. They typically have simpler architectures with fewer parameters, resulting in faster inference times and lower deployment costs. For example, DistilBERT [30] achieves 97% of BERT’s performance with 40% fewer parameters and 60% faster inference.

0.2.2.2 Speech Processing Specialized audio and speech processing systems provide essential capabilities for voice applications. Automatic Speech Recognition (ASR) systems like Whisper [34] convert spoken language to text with high accuracy across multiple languages. Text-to-Speech (TTS) systems generate natural-sounding speech from text input, with recent neural approaches achieving remarkable naturalness [42].

Voice activity detection, speaker diarization, and emotion recognition represent specialized audio processing capabilities that complement transcription, enabling richer understanding of speech content. These modular capabilities can be composed to build sophisticated applications without requiring end-to-end foundation models.

In the audio domain, specialized systems have shown significant advantages. MusicLM [1] represents a breakthrough in music generation from text descriptions, generating high-fidelity musical content that remains consistent over extended durations while following textual instructions. Agostinelli et al. [1] demonstrated that by focusing exclusively on music generation rather than general audio synthesis, MusicLM achieves remarkable quality and control in translating text descriptions into musical compositions. The specialized architecture employs a hierarchical sequence-to-sequence model that captures both local and global musical structures, enabling coherent compositions spanning multiple minutes - a capability that general-purpose models struggle to achieve. This focused approach allows MusicLM to incorporate domain-specific knowledge about musical theory, structure, and aesthetics that would be diluted in a general-purpose system.

0.2.2.3 Image and Video Processing Computer vision services enable applications to understand and process visual information. Object detection systems identify and locate objects within images [53], while image classification assigns categorical labels to images based on their content [52]. Face detection and recognition technologies identify the presence, location, and identity of faces in images [26], although their deployment raises significant privacy and ethical concerns.

Specialized video processing capabilities include action recognition [15], video summarization [12], and object tracking across frames [18]. These technologies enable applications ranging from surveillance systems to content recommendation engines.

The field of image generation has seen particularly rapid advancement in specialized models. Beyond DALL-E [38], which pioneered hierarchical text-conditional image generation with CLIP latents, models like Stable Diffusion [39] have demonstrated how latent diffusion models can generate high-resolution images with remarkable quality and efficiency. Rombach et al. [39] introduced a novel approach that performs diffusion in the latent space rather than pixel space, dramatically reducing computational requirements while maintaining generation quality. This architectural innovation makes high-quality image synthesis accessible to a much broader range of researchers and developers, as it can run on consumer-grade hardware rather than requiring specialized infrastructure. Stable Diffusion’s focus on latent diffusion represents a significant architectural specialization that optimizes specifically for the image generation task, demonstrating how domain-focused innovation can yield both quality and efficiency improvements beyond what general-purpose approaches might achieve.

Similarly, DALL-E’s predecessor, DALL-E 1 [38], established the groundwork for zero-shot text-to-image generation. Ramesh et al. [38] demonstrated that models could generate novel visual concepts from purely textual descriptions without domain-specific training. Their work showed that a specialized architecture designed specifically for cross-modal generation could produce images for text descriptions not seen during training, establishing a foundation for subsequent innovations in text-to-image synthesis. This specialized ap-

proach focused exclusively on the text-to-image task enabled architectural decisions optimized specifically for visual synthesis from linguistic descriptions.

0.2.2.4 Multi-Modal Processing Specialized multi-modal systems process and relate information across different modalities, such as text and images. Vision-language models like CLIP [33] learn joint embeddings of images and text, enabling powerful zero-shot classification capabilities. Image captioning systems generate textual descriptions of image content [27], while visual question answering systems respond to questions about image contents [13].

Recent advancements in multi-modal understanding have demonstrated the power of specialized architectures designed specifically for cross-modal learning. BLIP [25] introduced a bootstrapping approach for language-image pre-training that unifies vision-language understanding and generation, achieving state-of-the-art performance across a range of vision-language tasks. Li et al. [25] developed this specialized architecture to address limitations in existing vision-language models, particularly their reliance on noisy web-scraped image-text pairs. BLIP’s novel approach employs a captioner to generate synthetic captions and a filter to remove noisy captions, creating a bootstrapping mechanism that continuously improves the quality of training data. This specialized focus on data quality, combined with a multimodal mixture of encoder-decoder architectures, enables BLIP to outperform existing models on both understanding tasks (like visual question answering) and generation tasks (like image captioning). The architecture’s ability to both understand and generate across modalities while maintaining computational efficiency demonstrates the advantages of purpose-built systems over general-purpose alternatives.

Similarly, Flamingo [54] established a new standard for few-shot learning in visual domains. Yuste et al. [54] developed a specialized architecture that effectively connects pre-trained vision-only and language-only models through a lightweight cross-attention layer, enabling few-shot learning without requiring end-to-end retraining. This approach allows Flamingo to leverage existing pre-trained models while adding specialized components designed specifically for cross-modal few-shot learning. The architecture employs a unique “perceiver resampler” that efficiently processes visual information before integrating it with the language model, creating a modular design that maintains the strengths of both components. This specialized approach to visual language modeling enables remarkable few-shot learning capabilities across diverse visual tasks, including image captioning, visual question answering, and even video understanding, while requiring significantly fewer computational resources than end-to-end training of comparable models. Flamingo demonstrates how targeted architectural specialization focused on cross-modal interaction can achieve superior performance with greater parameter efficiency than monolithic alternatives.

Multi-modal foundation models like GPT-4V [45] and Gemini [10] have recently achieved impressive capabilities in processing mixed-modality inputs. However, these systems remain computationally intensive and often less reliable than specialized alternatives for specific multi-modal tasks. The performance gap between specialized multi-modal systems and general-purpose models reinforces the argument that architectural specialization offers significant advantages for tasks that require deep understanding of relationships between different information modalities.

0.2.2.5 Advantages of Specialization Specialized AI services offer several advantages over monolithic approaches:

1. **Performance and Efficiency:** Specialized systems generally achieve higher accuracy within their domains while requiring fewer computational resources. For example, specialized named entity recognition models consistently outperform general foundation models on domain-specific entity extraction tasks [23].
2. **Reliability and Determinism:** Domain-specific services typically exhibit more predictable behavior and deterministic outputs, making them better suited for critical applications where reliability is essential.
3. **Interpretability and Transparency:** Specialized systems often employ simpler architectures that facilitate better understanding of their decision-making processes. This transparency is particularly valuable in regulated industries and high-risk applications [43].
4. **Targeted Optimization:** Specialized services can be optimized for specific metrics relevant to their domains, whereas foundation models must balance performance across diverse tasks, leading to compromises.
5. **Deployment Flexibility:** Smaller specialized models can be deployed in diverse environments, including edge devices with limited computational resources, enabling AI capabilities in contexts where foundation models remain impractical.
6. **Resource-Efficient Task Adaptation:** Specialized models can be more efficiently adapted to specific tasks through techniques like task-aware distillation. Zheng et al. [55] demonstrated that task-aware layer-wise distillation can dramatically reduce model size while maintaining performance on specific tasks. Their “Less is More” approach showed that by selectively distilling knowledge from specific layers of larger models, smaller task-specific models could achieve 97% of the performance of much larger models while requiring only a fraction of the computational resources. This approach to efficient specialization provides another compelling example of how focused, task-specific architectures can offer significant efficiency advantages over general-purpose approaches.

These advantages create a compelling case for incorporating specialized services alongside foundation models in comprehensive AI systems. The challenge lies in effectively orchestrating these diverse components to leverage their respective strengths while presenting a unified interface to users and developers.

0.2.3 2.3 Orchestration Approaches and API Standardization Efforts

As AI capabilities have expanded, researchers and practitioners have developed various approaches to orchestrating multiple models and tools to address complex tasks. Simultaneously, efforts to standardize access to diverse AI services have emerged to address the fragmentation of provider interfaces. This section examines these orchestration approaches and API standardization efforts, along with their limitations relative to the goals of the current research.

0.2.3.1 Retrieval-Augmented Generation Retrieval-Augmented Generation (RAG) [24] represents one of the most successful and widely adopted approaches to extending foundation model capabilities. RAG combines parametric knowledge (information encoded in

model weights) with non-parametric knowledge (information retrieved from external sources at inference time). This approach enables foundation models to access up-to-date information, reference specific documents, and reduce hallucinations.

In their seminal work, Lewis et al. [24] demonstrated that combining dense passage retrieval with sequence-to-sequence models could dramatically improve performance on knowledge-intensive tasks. Their approach showed that retrieval components could effectively complement generative capabilities, achieving state-of-the-art results on question answering and fact verification benchmarks while maintaining the flexibility of generative models.

The RAG process typically involves: 1. Converting a user query into a search query 2. Retrieving relevant documents or passages from a vector database 3. Incorporating the retrieved information into the prompt provided to the foundation model 4. Generating a response that integrates both the model’s parametric knowledge and the retrieved information

Subsequent work has extended and refined the RAG approach, with systems like REALM [14], RETRO [6], and Atlas [19] demonstrating increasingly sophisticated retrieval-generation integration. REALM [14], developed by Guu et al., implemented a novel approach where the retriever and language model were jointly trained, allowing the system to learn retrieval patterns optimized for language modeling performance. This tight integration demonstrated how specialized retrieval components could be optimized specifically for language model augmentation. The key innovation in REALM was its end-to-end training methodology that updated both the neural retriever and language model simultaneously using a latent variable approach, enabling the system to learn retrieval patterns that directly improved language modeling performance rather than optimizing for traditional IR metrics. This joint training created a synergistic system where the retriever specifically learned to find passages that would help the language model’s predictions, regardless of whether those passages would be considered optimal by traditional information retrieval standards. Guu et al. [14] demonstrated that this approach significantly improved performance on knowledge-intensive tasks by creating a retriever specifically optimized for language model augmentation rather than general document retrieval.

Borgeaud et al. [6] introduced RETRO (Retrieval-Enhanced Transformer), which achieved superior performance by retrieving from trillions of tokens. Their approach showed that retrieval-enhanced models could match the performance of much larger parametric-only models while using significantly fewer parameters, highlighting the efficiency advantages of specialized retrieval components. RETRO’s architecture incorporated retrieval directly into the transformer architecture through specialized “retrieval blocks” interspersed between standard transformer layers. These blocks retrieved similar chunks from a massive database and conditioned the model’s predictions on both the retrieved examples and their similarity scores, creating a more tightly integrated approach than previous methods that simply prepended retrieved content to the input. The system demonstrated impressive scaling properties, with benefits increasing as more retrieval chunks were incorporated, suggesting that retrieval-augmentation represented a complementary scaling dimension to parameter scaling. The authors showed that a RETRO model with 7.5B parameters could match or exceed the performance of a conventional language model with 175B parameters on several benchmarks, demonstrating that retrieval could effectively substitute for significant parameter scaling in many applications.

More recently, Izacard et al. [19] developed Atlas, which applied few-shot learning tech-

niques to retrieval-augmented language models. By formulating diverse tasks in a consistent retrieval-followed-by-generation pattern, Atlas demonstrated remarkable cross-task transfer abilities, achieving strong performance on unseen tasks with minimal examples. This approach highlighted how retrieval augmentation enables more flexible adaptation to new domains. Atlas incorporated innovations in both architecture and training methodology, using a fusion-in-decoder approach that processed multiple retrieved passages in parallel before integrating their representations for generation. This architecture allowed the model to effectively reason across multiple retrieved documents, comparing and synthesizing information rather than treating each passage independently. The model was trained using a mixture of tasks formulated in a consistent retrieve-then-generate pattern, enabling it to develop general capabilities for leveraging retrieved information across diverse applications. This consistent formulation allowed Atlas to achieve remarkable transfer performance to new tasks not seen during training, demonstrating the potential for retrieval-augmentation to enhance model adaptability beyond just factual augmentation. The system’s strong performance with minimal examples on unseen tasks suggested that retrieval-augmentation could enhance few-shot learning capabilities by providing relevant context that reduces the need for extensive in-context examples or parameter updates.

While effective at enhancing foundation model capabilities, these RAG systems still maintain a clear separation between retrieval and generation components. The retrieval system remains subordinate to the language model, acting as an information provider rather than a peer in a collaborative system. This subordination differs fundamentally from the routing-based approach explored in this thesis, where specialized components are treated as first-class participants rather than augmentations.

Furthermore, current RAG implementations typically focus on text-to-text applications, with limited exploration of cross-modal retrieval-augmentation. The challenges of integrating retrieval across text, image, and audio modalities remain largely unexplored, representing a significant opportunity for future research in specialized model orchestration.

These systems effectively orchestrate the interaction between retrieval and generation components but focus primarily on augmenting a single foundation model rather than routing between specialized models. The success of RAG in reducing hallucinations and enhancing factuality nevertheless demonstrates the value of specialized components working in concert, providing empirical support for the potential of more sophisticated orchestration approaches like those explored in this thesis.

0.2.3.2 Tool-Using Agents Recent approaches have extended foundation models with the ability to use external tools. In these systems, a foundation model serves as a controller that decides when and how to invoke specialized capabilities. For example, Park et al. [53] demonstrated how large language models can be augmented to use Python to solve math problems, while Nakano et al. [53] showed how models can be trained to use web browsers.

Notable approaches in this category include:

1. **ReAct** [53]: Interleaves reasoning and acting, allowing the model to generate reasoning traces that explain its decision-making process and inform subsequent actions. By combining verbalized reasoning with action steps, ReAct enables more systematic and interpretable tool use compared to approaches that separate these processes.
2. **ToolFormer** [52]: Fine-tunes language models to use a range of external tools specified through API calls embedded in the model’s text generation. This self-supervised

approach allows the model to learn when and how to call external tools through in-context examples, reducing the need for manual prompt engineering.

3. **Function Calling:** Models like GPT-4 and Claude support structured function calling, allowing them to interact with external tools through well-defined interfaces [17]. This enables more reliable parsing of model outputs and integration with existing software systems. Recent standardization efforts like the Model Context Protocol [17] have attempted to formalize how LLMs communicate with external tools, introducing structured schemas and protocols for reliable function execution.

These tool-augmentation approaches represent significant advances but typically rely on reactive patterns where tools are invoked in response to specific problems. LangChain [9] has emerged as one of the most popular frameworks for building applications that leverage language models with tool usage capabilities. Chihab [9] developed LangChain as a composable framework that enables developers to create complex chains of operations involving language models, external tools, and structured data sources. The framework standardizes several key components needed for sophisticated LLM applications: prompts management, model interfaces, memory systems for maintaining context, and tool integration. LangChain’s architectural approach centers on the concept of “chains” - sequences of operations that can include language model calls, tool usage, memory operations, and output parsing. By standardizing these components with consistent interfaces, LangChain enables more systematic composition of capabilities than ad-hoc implementations. The framework has evolved to support “agents” that can reason about which tools to use and when, implementing various agent architectures including ReAct and structured output approaches. While LangChain provides significant scaffolding for tool integration, its fundamental approach still treats the language model as the central controller with tools as subordinate components, differentiating it from the more distributed architecture explored in Pseudo.

More sophisticated frameworks like BabyAGI [29] implement planning systems that use AI to create, prioritize, and execute tasks, demonstrating more proactive tool orchestration. Nakajima [29] introduced BabyAGI as a task management system that combines language models with vector storage to create, prioritize, and execute tasks autonomously. The system implements a task execution loop that continually generates new tasks based on previously completed ones and their outcomes, using a vector store to maintain context and prioritization mechanisms to determine execution order. This autonomous task management represents a significant advance in agent orchestration, enabling the system to break down complex objectives into manageable tasks without human intervention at each step. BabyAGI’s architecture demonstrates how language models can be structured to perform not just individual tasks but to manage entire workflows, establishing a pattern for autonomous systems that self-direct across multiple steps and tools. The system’s ability to maintain context across multiple execution cycles and adapt task priorities based on outcomes represents a more sophisticated orchestration approach than simple tool invocation.

Similarly, recent decoding techniques for LLMs [18] have improved the reliability of tool use through specialized sampling methods that enhance the model’s ability to generate syntactically valid tool invocations. These techniques include constrained decoding that enforces structural requirements on outputs, reducing the likelihood of syntax errors in tool calls and improving the overall reliability of agent-tool interactions.

These approaches represent significant advances in AI orchestration but typically maintain a foundation model as the central controller, with external tools serving as augmentations rather than equal participants in a distributed architecture. This controller-centric

design fundamentally differs from the routing-based approach proposed in this thesis, which treats specialized models as primary components rather than subordinate tools.

0.2.3.3 Multi-Agent Collaboration Multi-agent systems distribute complex tasks across specialized agents that collaborate to achieve common goals. Recent work has explored how multiple language model instances can collaborate to solve problems more effectively than a single model working in isolation [53, Patil and Zhang [31]].

Systems like AutoGen [50] and MetaGPT [31] provide frameworks for creating multi-agent systems where specialized agents engage in structured dialogues to address complex tasks. These approaches have demonstrated improvements in complex reasoning tasks, creative endeavors, and software development processes.

AutoGPT [50], developed by Weng [50], represents a significant advance in autonomous agent architectures. The system implements a goal-directed autonomous agent that can plan and execute complex sequences of actions to accomplish user-specified objectives. AutoGPT diverges from traditional tool-using agent designs by implementing a recursive goal refinement approach where the system breaks down high-level goals into actionable sub-goals, essentially becoming its own task manager. The architecture implements a cognitive loop that includes memory management (through vector storage), planning, and execution phases, with continuous self-assessment to determine when goals have been achieved or require refinement. This self-direction enables AutoGPT to navigate complex problem spaces with minimal human intervention, attempting to approximate artificial general intelligence through emergent planning capabilities. Unlike systems that require detailed instructions for each step, AutoGPT can operate with high-level directives and determine appropriate actions autonomously, representing a more sophisticated approach to agent orchestration than prompt-response models.

MetaGPT [31] introduces a particularly relevant approach that applies the software engineering paradigm to multi-agent coordination. By implementing a standardized operating procedure based on professional software development practices, MetaGPT structures multi-agent collaboration into discrete phases including requirement analysis, design, and implementation. This systematic workflow demonstrates how specialized agents with distinct roles can collectively tackle complex software engineering problems more effectively than individual models, aligning with the specialization principles explored in this thesis. The system leverages meta-programming techniques where agents produce structured outputs (like UML diagrams and task decompositions) that other agents can then process, creating a more organized collaborative workflow than unstructured dialogue-based approaches. Patil and Zhang [31] designed MetaGPT with explicit role-based specialization where different agents take responsibility for distinct aspects of the software development process - requirements analysis, architecture design, coding, testing, and review. Each agent's prompts and behaviors are specialized for its particular role, enabling more focused expertise than would be possible with a single general-purpose agent. This role specialization demonstrates how decomposing complex tasks across specialized components can yield superior results, providing empirical support for the specialization philosophy that also underpins Pseudo's approach.

However, current multi-agent approaches primarily employ homogeneous agents based on the same foundation model, rather than truly heterogeneous systems integrating fundamentally different AI technologies. They also typically lack robust routing mechanisms to direct tasks to the most appropriate specialized agents. This limitation highlights a significant gap addressed by the current research, which focuses specifically on intelligent routing

across diverse specialized models with different underlying architectures and capabilities.

0.2.3.4 API Abstraction Layers The fragmentation problem encountered during Pseudo’s development is not unique to this project, but rather represents a systemic challenge within the rapidly evolving AI ecosystem. As the landscape of AI services has proliferated, numerous abstraction layers have emerged to address the inconsistencies between provider interfaces. These solutions vary significantly in their approach, scope, and underlying architectural philosophies, each representing different trade-offs in the abstraction space.

LiteLLM [22] has gained significant adoption by providing an OpenAI-compatible interface for a wide range of LLM providers. By mimicking OpenAI’s request and response formats, LiteLLM enables developers to use consistent patterns across different providers, requiring minimal code changes when switching between models. This approach has several advantages: it leverages the widespread familiarity with OpenAI’s interface patterns, offers a gentle migration path for existing applications, and provides immediate utility without requiring wholesale application redesign.

However, LiteLLM’s design decision to adopt OpenAI’s patterns as the canonical interface creates certain limitations. The OpenAI interface was designed specifically for their models and services, embedding assumptions that may not align with the capabilities or optimal usage patterns of other providers. This “OpenAI-centricity” can lead to impedance mismatches when working with models that offer capabilities or parameters not present in OpenAI’s offerings. For example, Anthropic’s Claude models offer constitutional AI parameters [2] that have no direct equivalent in the OpenAI interface, requiring workarounds to utilize fully through an OpenAI-compatible abstraction.

Furthermore, LiteLLM’s focus remains primarily on text generation, with limited support for multimodal capabilities. While text generation represents a significant portion of current AI service usage, the increasing importance of multimodal models and specialized modality processors highlights the need for abstractions that can elegantly handle diverse output types and modality transformations. LiteLLM’s architecture was not fundamentally designed for this multimodal future, though it has added capabilities in this direction as these services have emerged.

Haystack [24] takes a fundamentally different approach to abstraction, offering a comprehensive framework for building natural language processing pipelines with modular components. Rather than simply wrapping provider APIs, Haystack introduces a more structured architecture based on readers, retrievers, and pipelines. This component-based approach emphasizes flexibility and composability, allowing developers to construct sophisticated workflows that combine multiple models and processing steps.

Haystack’s architecture offers significant advantages for certain types of applications, particularly those focused on retrieval-augmented generation (RAG) and question-answering tasks. Its structured pipeline approach enforces cleaner separation of concerns and enables more sophisticated composition of capabilities than simple API wrappers. However, this increased architectural sophistication comes with a steeper learning curve and potentially higher integration costs for simple use cases. Additionally, while Haystack supports multiple modalities in theory, its primary focus and optimization remain on text processing workflows, with less emphasis on general-purpose multimodal abstraction.

LangChain [9] has emerged as perhaps the most widely adopted framework in this space, offering comprehensive capabilities for chaining LLM operations with external tools and data sources. Its component architecture includes abstractions for various LLM providers,

but these abstractions exist primarily as elements within a broader framework rather than as independent utilities. LangChain’s emphasis on agent behaviors, tool integration, and complex conversational workflows addresses different aspects of AI development than pure provider abstraction.

While LangChain offers significant flexibility and power, its comprehensive scope can introduce complexity that may be unnecessary for applications simply seeking provider abstraction. The framework’s architectural decisions prioritize tool integration and agent behaviors, which may not align with the requirements of applications focused on direct model interactions. Furthermore, LangChain’s approach to provider abstraction has evolved alongside its broader feature set, sometimes leading to inconsistencies or limitations in the abstraction layer itself.

LlamaIndex [42] provides a data framework specifically designed for LLM applications, focusing on retrieval-augmented generation and data integration. Developed by Shivakumar et al. [42], LlamaIndex differs from provider abstraction layers by focusing on the data layer rather than the model layer. The framework provides tools for indexing, querying, and retrieving information from diverse data sources, creating structured representations that can be effectively utilized by language models. LlamaIndex’s approach complements provider abstraction by addressing how models interact with external data, implementing specialized data structures and retrieval mechanisms optimized for language model consumption. The framework’s emphasis on data integration demonstrates how the AI ecosystem requires standardization at multiple levels - not just provider interfaces but also data interfaces and processing pipelines. LlamaIndex’s architecture includes modular components for data connectors, indices, retrievers, and query engines, enabling flexible composition similar to the componentized approach taken by APICenter. While addressing a different aspect of the fragmentation problem, LlamaIndex’s approach reinforces the importance of standardized interfaces between components in complex AI systems.

TorchServe [28] offers a flexible and standardized model serving solution for PyTorch models, addressing deployment standardization rather than API standardization. McIntosh et al. [28] developed TorchServe to provide a consistent deployment mechanism for diverse PyTorch models, regardless of their architecture or purpose. The system implements a handler-based architecture that adapts different model types to a consistent serving interface, enabling standardized model management and inference. TorchServe’s focus on operational aspects of model deployment - scaling, monitoring, versioning, and inference optimization - addresses challenges that higher-level abstractions often overlook. The system’s design acknowledges that deployment considerations significantly impact the practical usability of AI models, especially in production environments. By providing a standardized serving layer, TorchServe creates foundation infrastructure upon which higher-level abstractions like APICenter can build, demonstrating the complementary nature of standardization efforts at different levels of the AI stack.

Model Context Protocol [17], developed by Hou et al. [17] at Anthropic, represents a more specialized standardization effort focused specifically on communications between language models and tools. The protocol establishes structured schemas for tool definitions, invocations, and responses, enabling more reliable parsing and execution of tool calls from language models. Unlike provider abstraction layers that focus on accessing different models, Model Context Protocol focuses on standardizing how models interact with multiple tools. This standardization addresses a critical pain point in tool-using agent architectures: the inconsistency and unreliability of tool invocations from language models. The protocol’s emphasis on structured interfaces between components aligns with the separation of

concerns philosophy that underlies the Pseudo/APICenter architecture, demonstrating how standardization can enable more reliable interactions between specialized components in an AI system.

These diverse approaches demonstrate how standardization efforts can target different layers of the AI stack, from model serving to API interfaces to application frameworks. Each approach addresses different aspects of the fragmentation challenge, with varying tradeoffs between simplicity, comprehensiveness, and flexibility.

Comparing these approaches reveals several patterns in the abstraction space. Simplicity often trades off against comprehensiveness, with simpler abstractions like LiteLLM offering lower barriers to entry but potentially limiting the expression of provider-specific capabilities. Architectural sophistication can enable more complex workflows but may increase integration complexity and learning curves. Provider-centric designs (like LiteLLM’s OpenAI compatibility) simplify migration from specific providers but may constrain the expression of capabilities from other providers.

APICenter’s design represents a different point in this trade-off space, emphasizing modality-based abstraction rather than provider-based patterns. By organizing its interfaces around modalities (text, image, audio) rather than specific provider patterns, APICenter creates an abstraction that can more naturally express the capabilities of diverse providers without privileging any particular provider’s interface design. This approach requires developers to learn APICenter’s interface patterns rather than leveraging existing familiarity with specific provider SDKs, but offers greater flexibility and expressiveness across the provider ecosystem.

The modality-centric approach aligns more closely with the underlying capabilities being abstracted rather than the specific implementations of those capabilities by particular providers. This alignment enables more natural expression of provider-specific extensions while maintaining consistency at the modality level. For example, text generation interfaces can include common parameters like temperature and maximum length shared across all providers, while still allowing provider-specific extensions for unique capabilities like constitutional AI parameters or specialized inference options.

APICenter also differs from existing solutions in its focus on completeness across modalities rather than depth in specific workflows. While frameworks like LangChain offer sophisticated capabilities for specific types of applications (particularly agent-based systems), APICenter prioritizes comprehensive coverage of modalities and providers with a simpler, more focused abstraction model. This approach aligns more closely with Pseudo’s requirements for a universal routing system capable of handling diverse input and output types across multiple providers.

0.2.3.5 Limitations of Current Approaches Existing orchestration and standardization approaches exhibit several limitations relevant to the goals of the current research:

1. **Foundation Model Centricity:** Most approaches position a foundation model as the central coordinator, with specialized tools serving as augmentations rather than equal participants in a distributed architecture.
2. **Limited Modality Support:** Current standards primarily address text generation, with less developed support for multi-modal inputs and outputs spanning text, images, audio, and video.

3. **Manual Integration:** Developers must typically perform significant integration work to combine diverse AI services, with limited automation in service selection, request routing, and result processing.
4. **Weak Classification:** Existing systems lack robust classification capabilities for identifying the most appropriate service for a given request based on task characteristics, modality requirements, and other factors.
5. **Provider Heterogeneity:** Despite standardization efforts, significant heterogeneity persists in provider interfaces, authentication mechanisms, rate limiting, and other operational aspects.

The approach described in subsequent sections addresses these limitations through a classification-first architecture that routes requests to specialized services based on comprehensive task analysis.

0.3 Chapter 3: Methods

This chapter provides a comprehensive exposition of the methodological approach employed in the development of Pseudo and APICenter. Rather than accepting the prevailing dichotomy between monolithic omni-models and fragmented specialized tools, the system architecture presented here establishes a third path—one that combines the seamless user experience of omni-models with the superior performance and efficiency of specialized models. The chapter begins with an overview of this novel architectural approach, explaining the rationale behind the two-component design and the separation of concerns. It then examines in detail the implementation of APICenter as a unified provider interface, followed by an exploration of Pseudo’s intelligent content routing system. Throughout the chapter, particular attention is paid to the technical challenges encountered during development and the solutions implemented to address them.

0.3.1 3.1 Architecture and Design Principles

The development of Pseudo and APICenter follows a systems-thinking approach that challenges the fundamental assumptions underlying both dominant AI paradigms. Rather than accepting the tradeoff between convenience and specialized excellence, this architecture demonstrates how intelligent routing and standardized abstraction can achieve the benefits of both approaches simultaneously. This section details the architectural decisions that shaped these complementary systems and the rationale behind their design.

The solution presented in this thesis implements a deliberate separation between provider abstraction (APICenter) and intelligent routing (Pseudo). This architectural decision follows established principles in software design, particularly the single responsibility principle articulated by Martin [27], which advocates that each component should have only one reason to change. By separating these concerns, the architecture achieves greater modularity, reusability, and maintainability than would be possible with a monolithic approach.

The system architecture draws extensively from established software engineering patterns articulated by Gamma et al. [13]. The Adapter pattern is applied consistently throughout APICenter to translate between the standardized interface and provider-specific APIs. This pattern enables the system to “convert the interface of a class into another interface clients expect,” allowing fundamentally different provider interfaces to be accessed through a consistent mechanism. Gamma et al. [13] identified the Adapter pattern as particularly valuable

in systems that must integrate with existing interfaces that cannot be modified directly – precisely the situation faced with proprietary AI provider APIs. The implementation follows their recommended structure with adapter classes that map between the target interface (APICenter’s standardized interface) and the adaptee interfaces (provider-specific APIs). Similarly, the Facade pattern provides a simplified interface to the complex subsystem of provider integrations, hiding implementation details and reducing coupling between client code and specific providers. These established design patterns provide proven solutions to recurring design challenges, creating a more maintainable and extensible architecture than would be possible with ad-hoc solutions.

The architecture also draws concepts from distributed systems theory as formalized by Tanenbaum and Van Steen [45]. Their principles on masking failures in distributed environments informed APICenter’s error handling approach, which isolates clients from the heterogeneous error reporting mechanisms of different providers. The architecture implements what Tanenbaum and Van Steen [45] term “transparent distribution” – hiding the complexities of interacting with distributed components (in this case, diverse AI provider services) behind a unified interface that masks the underlying heterogeneity. This transparency extends to failure handling, where provider-specific errors are mapped to a consistent exception hierarchy that maintains essential information while providing uniform handling patterns. These distributed systems principles are particularly relevant given the inherently distributed nature of AI service consumption, where requests must traverse network boundaries to reach provider-hosted models with varying reliability characteristics.

The two-component design implements the broader architectural principle of separation of concerns, which Stevens et al. [43] identified as essential for managing complexity in software systems. Their foundational work on structured design established that systems should be organized to minimize coupling between components while maximizing the cohesion within each component. Pseudo and APICenter reflect this principle through their clear functional boundaries and well-defined interfaces between components. Stevens et al. [43] introduced metrics for coupling and cohesion that have become standard tools for evaluating software architectures – by these metrics, the separation of provider abstraction from intelligent routing creates higher internal cohesion within each component (as measured by the relatedness of functions within each module) while reducing coupling between them (as measured by the interdependence between modules). This architectural approach aligns with their guidance that “good systems can be expected to have relatively high coherence within a module and relatively loose coupling between modules,” creating a more maintainable and evolvable system structure.

Figure 3.1: Two-Component Architecture



Content Analysis
& Classification

Provider 1
(e.g., OpenAI)

Provider Selection
& Orchestration

Provider 2
(e.g., Anthropic)

Response Normalization
& Presentation

Provider n
(e.g., Stability AI)

This architecture implements a clear separation between:

1. **Pseudo (Intelligence Layer)**: Responsible for content analysis, modality classification, provider selection, and orchestration logic. This component embodies the core research contribution of intelligent routing based on content analysis.
2. **APICenter (Abstraction Layer)**: Provides a standardized interface to diverse AI providers across modalities. This component solves the “provider abstraction problem” identified in Chapter 2, enabling unified access to text, image, and audio generation services.

The relationship between these components is unidirectional—Pseudo depends on APICenter but not vice versa. This architectural decision ensures that APICenter can function as a standalone library for developers seeking standardized provider access without the routing capabilities of Pseudo. Meanwhile, Pseudo can leverage the abstraction provided by APICenter to focus exclusively on its core responsibility of intelligent routing without concern for the implementation details of specific provider APIs.

This unidirectional dependency structure follows what Gamma et al. [13] term the “Dependency Inversion Principle,” where high-level modules depend on abstractions rather than concrete implementations of lower-level modules. By depending on APICenter’s abstract interface rather than specific provider implementations, Pseudo achieves greater flexibility and adaptability to changes in the provider ecosystem. This design approach follows the established pattern of layered architectures described by Buschmann et al. in their extension of design patterns work, creating a clean separation of concerns between the application logic layer (Pseudo) and the infrastructure abstraction layer (APICenter).

Together, these components implement a system that fundamentally challenges the prevailing assumption that users must choose between the convenience of omni-models and the

excellence of specialized models. By dynamically routing requests to appropriate specialized models while presenting a unified interface to users, the system effectively simulates omni-model behavior without the limitations of monolithic architectures.

In practice, this architectural approach is implemented through two distinct but complementary tools:

1. Pseudo: A Complete End-User System

Pseudo is implemented as a Flask web application with a Python backend and a vanilla JavaScript frontend. It provides users with a complete interface for interacting with AI capabilities across multiple modalities:

- **Modern Chat Interface:** A clean, user-friendly interface resembling popular chat applications, allowing users to interact with the system using natural language
- **Multi-Modal Response Handling:** Seamless display of text, images, and audio within the same conversation flow
- **Conversation Management:** Persistent storage of chat history with the ability to browse past conversations
- **Media Organization:** Automatic storage and management of generated images and audio files

The backend architecture of Pseudo consists of several key components:

- **Core Application (app.py):** Flask application setup and entry point
- **Config Management (config.py):** Configuration loading and environment variable handling
- **Routes (routes.py):** API endpoints and HTTP request handling
- **Content Router (content_router.py):** Routes user inputs to appropriate AI providers
- **Chat History Manager (chat_history.py):** Manages persistent chat storage
- **Media Manager (media_manager.py):** Handles media file operations

This structure enables Pseudo to present a unified interface to users while intelligently orchestrating specialized models behind the scenes, effectively simulating the experience of interacting with an omni-model.

2. APICenter: A Developer Library for Provider Abstraction

APICenter, in contrast, is implemented as a Python library focused exclusively on standardizing interactions with diverse AI providers. It offers developers a consistent pattern for accessing AI capabilities across providers and modalities:

```
# Text generation with different providers using identical patterns
response_openai = apicenter.text(
    provider="openai",
    model="gpt-4",
    prompt="Write a poem about birds"
)

response_anthropic = apicenter.text(
    provider="anthropic",
    model="claude-3-opus-20240229",
    prompt="Write a poem about birds"
```

```

)

# Similar consistency for image generation
image_openai = apicenter.image(
    provider="openai",
    model="dall-e-3",
    prompt="A serene mountain lake at sunset"
)

image_stability = apicenter.image(
    provider="stability",
    model="stable-diffusion-xl-1024-v1-0",
    prompt="A serene mountain lake at sunset"
)

```

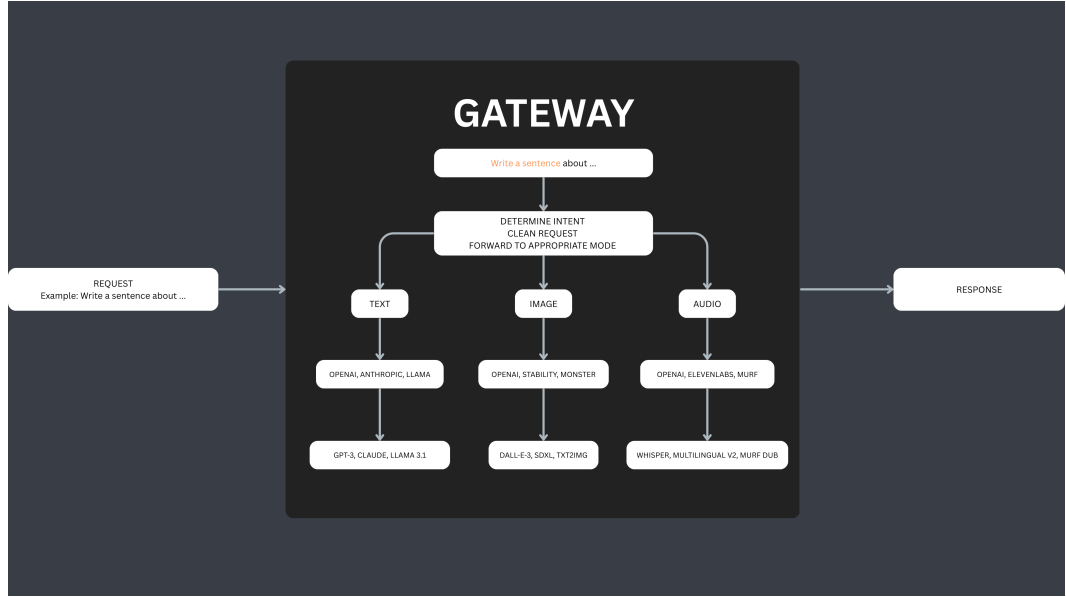
The library manages several complexities behind this simple interface:

- **Credential Management:** Secure handling of API keys and authentication for different providers
- **Parameter Mapping:** Translation between standardized parameters and provider-specific parameters
- **Input Format Conversion:** Transformation of unified input formats to provider-specific requirements
- **Response Normalization:** Standardization of diverse response formats into consistent structures
- **Error Handling:** Uniform error reporting across providers with different error formats

This standardization enables developers to write provider-agnostic code that can easily switch between different AI services or implement fallback mechanisms without requiring provider-specific logic.

The system implements a multi-stage data flow architecture that processes user requests through several distinct phases, maintaining separation of concerns while ensuring effective coordination between components.

Figure 3.2: Data Flow Architecture



= 100%}

This data flow architecture implements concepts from structured design as established by Stevens et al. [43], particularly the principle of “transform-centered” design that processes data through sequential transformations. Each processing stage represents what Stevens et al. [43] termed an “afferent-transform-efferent” pattern, where each component receives input (afferent), performs a transformation, and produces output (efferent) for the next component. This structured approach to data flow creates cleaner interfaces between components and facilitates reasoning about system behavior, as recommended in their foundational work on structured design.

The architecture also implements several principles from distributed systems as described by Tanenbaum and Van Steen [45], particularly:

1. **Progressive Transformation:** User requests undergo progressive transformation through successive processing stages, from natural language input to provider-specific API calls to standardized response formats.
2. **Information Enrichment:** Each processing stage adds contextual information to the request, including classification metadata, selected provider details, and transformation directives.
3. **Failure Isolation:** The staged architecture enables isolation of failures at specific points in the processing pipeline, facilitating graceful degradation and fallback strategies. This implementation follows what Tanenbaum and Van Steen [45] describe as “compartmentalization of failures” in distributed systems, where failures in one component are contained rather than propagating throughout the system.
4. **Asynchronous Processing:** The architecture supports both synchronous and asynchronous processing models, allowing for non-blocking handling of long-running operations. This flexibility aligns with distributed systems principles regarding communication patterns and timing models as described by Tanenbaum and Van Steen [45].

The data flow represented in Figure 3.2 illustrates the complete path from user request to response, highlighting the clear separation between the intelligence layer (Pseudo) and the abstraction layer (APICenter) while showing their coordinated interaction.

The implementation supports flexible deployment configurations to accommodate different usage scenarios, from personal use to multi-user environments. The architecture uses a three-tier structure with client layer (web browsers, external applications), application layer (Pseudo server, APICenter library), and provider layer (external AI services). This modular design ensures that components can be scaled independently based on usage patterns and resource requirements.

The implementation utilizes a carefully selected technical stack that balances modern capabilities with stability and accessibility:

1. **Backend Language:** Python 3.12, selected for its rich ecosystem of AI and web development libraries, readability, and widespread adoption in the AI community.
2. **Web Framework:** Flask, chosen for its lightweight nature, flexibility, and ease of integration with AI processing pipelines.
3. **Frontend Technologies:** Vanilla JavaScript, HTML5, and CSS3, selected to minimize dependencies while providing a responsive, modern user interface.
4. **Data Storage:** File-based storage using JSON for structured data and direct file storage for media assets, chosen for simplicity and portability.
5. **External Dependencies:** Carefully curated to include only essential libraries, with clear separation between core dependencies and optional extensions.

This technical stack prioritizes clarity, maintainability, and accessibility, enabling other researchers to easily understand, extend, and deploy the system for further experimentation.

0.3.2 3.2 APICenter: Provider Abstraction Implementation

APICenter represents the foundational layer of the two-component architecture, providing standardized access to diverse AI services across modalities. This section details the design principles, technical implementation, and key challenges addressed in the development of this unified provider interface.

APICenter’s development was guided by four core design principles that address the fundamental challenges identified in the provider abstraction gap discussed in Chapter 2:

Provider-Agnostic Interface

The primary design goal of APICenter is to offer a consistent method signature regardless of the underlying provider. This principle embodies the “write once, use everywhere” philosophy articulated by Gamma et al. [13] in their work on design patterns, particularly the Adapter pattern which “converts the interface of a class into another interface clients expect.” By implementing this pattern at scale across multiple providers and modalities, APICenter enables developers to write code once and deploy it across diverse AI services without modification.

The implementation follows a simple, intuitive pattern that is consistent across providers and modalities:

```

result = apicenter.<mode>(
    provider="<provider_name>",
    model="<model_name>",
    prompt="<input>",
    **kwargs
)

```

This consistency dramatically reduces the cognitive overhead for developers working with multiple AI services, allowing them to focus on their application logic rather than managing provider-specific implementation details.

Consistent Error Handling

Error management represents a significant challenge in cross-provider applications due to the diverse error reporting mechanisms employed by different providers. APICenter implements a standardized error handling approach inspired by Tanenbaum and Van Steen’s [45] principles for distributed systems, particularly their concept of “masking failures” to provide a consistent experience despite underlying heterogeneity. This approach translates provider-specific errors into a unified format that preserves essential information while enabling consistent handling.

Credential Management

Secure and flexible credential management forms another core principle of APICenter’s design. Drawing on Saltzer and Schroeder’s [40] principles of protection, particularly the concepts of “least privilege” and “separation of mechanism and policy,” APICenter implements a modular credential management system that separates credential storage from API implementation while ensuring appropriate access scoping.

APICenter looks for credentials in several standard locations: 1. Custom path specified by `APICENTER_CREDENTIALS_PATH` environment variable 2. Current working directory: `./credentials.json` 3. Project root directory: `<project_root>/credentials.json` 4. User’s home directory: `~/.apicenter/credentials.json` 5. System config directory: `~/.config/apicenter/credentials.json`

This flexible approach accommodates various deployment scenarios while maintaining security best practices.

Response Standardization

The final core principle addresses the challenge of diverse response formats across providers. Building on principles from data integration theory [23], APICenter implements a response standardization approach that preserves provider-specific metadata while presenting a consistent structure to consumers. This standardization simplifies post-processing and enables seamless provider switching without requiring code changes to handle different response formats.

APICenter is organized into a modular architecture with clear separation of concerns, following principles of cohesion and coupling that have been established best practices in software engineering since Stevens et al. [43]. Its component types include the public interface, core components, mode components, provider adapters, and type system & utilities. This modular organization follows the dependency inversion principle articulated by Martin [27], with high-level components depending on abstractions rather than concrete implementations of lower-level components.

The implementation of API standardization represents the core technical contribution of APICenter. The system implements a consistent method signature pattern across all

modalities that accommodates different input types while maintaining a consistent structure. The implementation uses Python’s flexible keyword argument system to accommodate provider-specific parameters without requiring changes to the core interface.

One of the most significant challenges in provider unification is managing inconsistent parameter naming and behavior. APICenter implements a parameter transformation system that maps standardized parameters to provider-specific equivalents. Each provider adapter follows a consistent pattern to encapsulate provider-specific logic, implementing four key transformations: client initialization, prompt formatting, response standardization, and error transformation. Together, these transformations entirely encapsulate provider-specific implementation details, allowing the rest of the system to interact with a clean, consistent interface.

APICenter implements specialized handling for each supported modality, addressing the unique challenges and requirements of text, image, and audio generation. The text generation mode supports diverse formats across providers, including simple completion models and chat-based models. The image generation mode addresses challenges like diverse output formats and parameter standardization. The audio mode presents unique challenges in voice selection and format handling. For each modality, APICenter provides consistent interfaces while handling the complex transformations required for different providers.

APICenter implements a robust credential management system that balances security, usability, and flexibility. The credential discovery process searches for configuration in a predefined order, and the credential structure follows a hierarchical organization by mode and provider. The credential loading process includes validation to provide clear feedback about configuration issues. The system also implements several security best practices for credential management.

The development of APICenter encountered several significant technical challenges, each requiring innovative solutions. These included inconsistent response structures across providers, provider-specific rate limiting, authentication diversity, and parameter equivalence mapping. The implementation addresses these challenges through flexible response standardization, adaptive rate limiting, a flexible authentication framework, and a comprehensive parameter transformation system.

APICenter’s reliability depends on thorough testing across diverse providers and scenarios, including mock-based unit testing, integration testing, parametrized testing, response verification, and error handling verification. The system also implements several performance optimization strategies, including connection pooling, response streaming, lazy initialization, caching, and parallel processing.

By providing a foundational abstraction layer that significantly reduces the complexity of working with multiple AI providers, APICenter enables developers to write provider-agnostic code, switch between providers without code changes, implement fallback strategies for enhanced reliability, and integrate diverse AI capabilities with minimal integration complexity.

0.3.3 3.3 Pseudo: Intelligent Routing System

Building upon the foundation provided by APICenter, Pseudo implements an intelligent content routing system that directs user requests to appropriate specialized models based on sophisticated content analysis. This section details the classification system that forms the core of Pseudo’s routing capabilities and the provider selection algorithm that determines optimal routing paths.

The classification system at the core of Pseudo determines the appropriate modality and specific requirements from natural language inputs. While the theoretical design could implement sophisticated hybrid approaches combining pattern-based techniques with semantic analysis, the current implementation takes a more practical approach by leveraging a large language model (LLM, a neural network trained on vast text corpora that can generate human-like text and perform various language tasks).

The current Pseudo implementation employs the deepseek-r1 model through Olama to perform content classification, as shown in the actual implementation of the `select_mode_and_clean_content` method:

```
def select_mode_and_clean_content(self, user_input: str) -> tuple[str, str]:
    """Determine content type and extract cleaned content from user input.

    Returns a tuple of (mode, cleaned_content) where mode is one of 'text', 'image', 'audio'
    and cleaned_content is the extracted actual content the user wants to process.
    """
    try:
        # System prompt for content classification and extraction
        system_prompt = """You are a content analyzer that determines both the type of content received
        and the actual content to be processed.

        Based on the user's input, you must respond with EXACTLY this format:

        mode: <mode>
        content: <cleaned content>

        Where <mode> is one of: 'text', 'image', or 'audio'
        And <cleaned content> is the actual content to be processed (removing meta-instructions).

        Rules for determining mode:
        1. For IMAGE requests - Detect when the user clearly wants a visual representation:
            - Look for explicit visual keywords: image, picture, photo, drawing, illustration, render, visualize, etc.
            - Consider requests about visually-oriented subjects: landscapes, scenes, portraits, designs, etc.

        2. For AUDIO requests - Strictly for text-to-speech conversion only:
            - IMPORTANT: Audio mode ONLY converts text to spoken speech, it cannot create music or sound effects
            - Only choose audio when the user explicitly wants text spoken aloud
            - Look for audio keywords: speak, read aloud, narrate, voice, speech, pronounce, recite, say, etc.
            - The intent should be to have specific text converted to spoken audio
            - MUST have clear indication of speech intent - just providing a quote or sentence is NOT enough

        3. For TEXT requests - This is the primary generative content mode:
            - Any requests for information, explanations, stories, essays, paragraphs
            - Content that is meant to be read rather than seen or heard
            - Requests using writing verbs: write, explain, tell me, describe, summarize, compose, etc.
            - Any ambiguous requests about creating content (when not clearly visual)
            - Standalone quotes or sentences without speech indicators should be treated as text
        """
```

```

4. For VAGUE requests - When the intent is unclear:
- If the request contains an equal mix of indicators, prefer text over other modes
- Words like "create," "generate," "make," "produce" without clear visual/audio context → text
- Requests about "sound" or "music" should be text, as audio is only for text-to-speech
- When truly ambiguous, default to text mode as it's the most general-purpose
- "Compose a scene" → text (unless explicitly asking for a visual)
- Standalone quotes or sentences → text (unless explicitly asked to be spoken)
"""

# Use queue-based approach from credentials.json - try providers in strict order
if "text" in self.credentials["modes"]:
    text_providers = self.credentials["modes"]["text"]["providers"]

    # Try each provider in the order they appear in credentials.json
    for provider_name, provider_config in text_providers.items():
        if "models" in provider_config and provider_config["models"]:
            # Use the first model in the list (queue order matters)
            model_name = provider_config["models"][0]

            try:
                logger.info(
                    f"Using {provider_name}/{model_name} for content detection and cleaning"
                )

                # Use apicenter singleton to make the classification and extraction
                response = self.api_center.text(
                    provider=provider_name,
                    model=model_name,
                    prompt=[
                        {"role": "system", "content": system_prompt},
                        {"role": "user", "content": user_input},
                    ],
                    temperature=0.0,
                )

                # Parse the response to extract mode and cleaned content
                # [parsing logic omitted for brevity]

                # Validate the extracted information
                if mode in ["text", "image", "audio"] and cleaned_content:
                    logger.info(
                        f"Mode detected: {mode}, Cleaned content: '{cleaned_content}'"
                    )
                    return mode, cleaned_content
            except Exception as e:
                # Error handling logic
                continue # Try next provider in queue

```

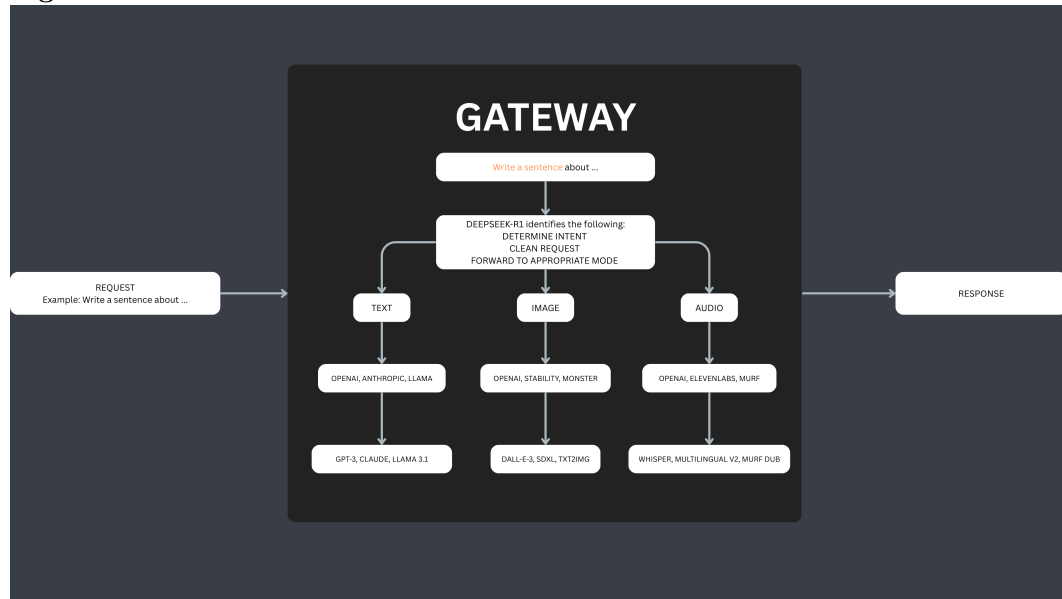
```

# Default fallback
return "text", user_input
except Exception as e:
    logger.error(f"Error in mode and content detection: {e}")
    return "text", user_input # Default to text mode with original input on error

```

This implementation relies on an extensive system prompt to instruct the language model on classification criteria. The prompt provides clear guidelines for distinguishing between different modalities based on linguistic cues and handles ambiguous requests by defaulting to text mode. This approach leverages the semantic understanding capabilities of the deepseek-r1 model, but in a more directed manner than fully autonomous classification.

Figure 3.5: Current LLM-Based Classification Flow



= 100%}

Theoretical Approaches to Classification

While the current implementation uses a straightforward LLM-based approach, more sophisticated classification architectures could be developed in future iterations. Drawing on established research in information retrieval and natural language processing [51, Lenzerini [23]], a hybrid approach combining multiple techniques could potentially offer advantages in accuracy and efficiency:

1. **Pattern-Based Classification:** A rule-based system using regular expressions to identify explicit modality indicators in user inputs. This approach would be particularly effective for unambiguous requests and could reduce computational overhead compared to full LLM inference. Lenzerini [23] has demonstrated that pattern-based approaches can provide high precision for structured classification tasks.
2. **Keyword Analysis:** Statistical analysis of modality-specific terminology frequency and distribution. This approach, aligned with Wei et al.'s [48] research on emergent abilities in language models, could provide an efficient intermediate layer for moderately explicit requests.

3. **Semantic Similarity:** Embedding-based comparison between user inputs and exemplars of different modalities. This technique adapts Borgeaud et al.’s [6] work on retrieval-augmented generation to the challenge of modality classification and could be particularly valuable for less explicit or novel request formulations.
4. **Machine Learning Classification:** A supervised classifier trained on labeled examples of different modality requests. Research by Wolf et al. [51] suggests that such classifiers can achieve high accuracy with relatively small training datasets when the classification boundaries are well-defined.

A potential implementation might integrate these approaches in a tiered architecture, starting with computationally efficient pattern matching for clear cases, progressing to keyword analysis for less clear cases, and finally using semantic similarity or the full LLM approach only for the most ambiguous requests. This would maintain the classification accuracy of the current approach while improving efficiency for common request types.

The gap between the current LLM-based implementation and these theoretical approaches represents a significant opportunity for future development. The current implementation provides a functioning proof of concept, while the theoretical approaches suggest promising directions for system evolution.

The current implementation of Pseudo uses a straightforward provider selection approach rather than the more sophisticated algorithms that could theoretically be developed. The system checks if the requested mode is configured in the credentials file, validates that valid provider configurations exist for the specified mode, tries providers in the exact order they appear in the credentials file, tries models in the order they are listed for each provider, and returns the first successful response immediately. If all attempts fail, the system returns an informative error message.

This implementation uses a basic sequential queue approach rather than sophisticated selection algorithms. The strict ordering of providers and models in the credentials file determines the selection priority, making the system’s behavior predictable and controllable through configuration changes.

While simple, this approach does provide a basic fallback mechanism that enhances reliability by trying alternative providers when the first choice fails. This can be viewed as a rudimentary implementation of the circuit breaker pattern described by Tanenbaum and Van Steen [45] for resilient distributed systems, though it lacks the state management and adaptive behavior of more sophisticated implementations.

Theoretical Enhancements

Looking beyond the current implementation, more sophisticated selection algorithms could significantly enhance the system’s intelligence and adaptability. These theoretical enhancements, informed by research in distributed systems [45] and recommendation systems [53], could include:

1. **Capability Profile Matching:** Maintaining detailed capability profiles for each provider/model combination and matching these against parsed request requirements. This approach would enable more intelligent selection based on the specific capabilities needed for a particular request.
2. **Performance-Based Selection:** Recording and analyzing performance metrics (response time, quality, error rates) to dynamically adjust provider preferences based on observed behavior. This would align with principles of online learning as described by Yao et al. [53].

3. **Cost Optimization:** Balancing quality requirements with operational costs based on provider pricing tiers and request complexity. This optimization could reduce overall operational costs while maintaining response quality.
4. **Adaptive Fallback Strategies:** Implementing more sophisticated fallback mechanisms that adapt based on failure patterns and provider reliability statistics, potentially skipping providers with recent failures.

The gap between the current straightforward implementation and these theoretical enhancements represents a significant opportunity for future development. While the current approach provides functional routing and basic reliability through sequential fallbacks, more sophisticated approaches could enhance both the intelligence and efficiency of the selection process.

User Experience and Media Handling

A critical aspect of Pseudo's design is the user experience layer that simulates the seamless interaction typically associated with monolithic omni-models. The system implements a comprehensive chat interface with multi-modal response handling:

Chat History System

Pseudo implements a sophisticated chat history system that provides persistent storage of all conversations, including text, images, and audio files:

```
chat_history/
  history.json          # Global index of all chats
  [chat-uuid]/         # Individual chat directory
    metadata.json      # Chat metadata and messages
    media/             # Media storage
      image_[timestamp].png # Image files
      audio_[timestamp].mp3 # Audio files
```

This structure ensures that all interactions are preserved, allowing users to revisit past conversations and access generated media. The system maintains a global index (`history.json`) that tracks minimal metadata for all chats, enabling fast loading in the sidebar:

```
{
  "chats": [
    {
      "id": "186be78d-b48f-4c9f-9216-f0a3a0336f4c",
      "title": "Generate an image of a cat",
      "created_at": "2025-04-02T03:58:00.466127",
      "updated_at": "2025-04-02T03:58:53.466127"
    },
    ... additional chats ...
  ]
}
```

Each individual chat has its own `metadata.json` file containing complete conversation details:

```

{
  "id": "186be78d-b48f-4c9f-9216-f0a3a0336f4c",
  "title": "Generate an image of a cat",
  "created_at": "2025-04-02T03:58:00.466127",
  "updated_at": "2025-04-02T03:58:53.466127",
  "messages": [
    {
      "role": "user",
      "content": "Generate an image of a cat",
      "timestamp": "2025-04-02T03:58:00.466127"
    },
    {
      "role": "assistant",
      "mode": "image",
      "content": "Generated content",
      "media": "image_20250402_035853.png",
      "provider": "openai",
      "model": "dall-e-3",
      "timestamp": "2025-04-02T03:58:53.466127"
    }
  ]
}

```

Multi-Modal Response Handling

Pseudo handles three types of content, each with specialized processing:

1. **Text Content:** Direct storage in the message objects within the chat's metadata
2. **Image Content:** Processing workflow that saves generated images to the chat's media directory and records references in the message object
3. **Audio Content:** Similar processing for audio files, with specialized handling for playback in the frontend

The frontend renders appropriate elements for each content type, creating a seamless experience regardless of the output modality. Images are displayed inline with loading states and error handling, while audio content is presented with player controls for convenient playback.

This comprehensive handling of diverse content types allows Pseudo to simulate the unified experience of an omni-model while leveraging specialized models behind the scenes. Users interact with a single, consistent interface regardless of whether they are generating text, images, or audio, with the underlying routing complexity completely hidden from view.

State Management and Error Handling

To maintain the illusion of a single unified system, Pseudo implements sophisticated state management and error handling:

State Management

The system maintains state through several mechanisms:

1. **Backend State:**
 - Chat Manager Context: Maintains chat session information

- Filesystem: Persistent storage of chat history and media
- In-Memory Caching: Performance optimization for frequent operations

2. Frontend State:

- Current Chat: Tracks the active chat ID and message history
- Temporary Chats: New chats start with a temporary ID until the first message is sent
- Chat ID Mapping: Maintains consistency between temporary and permanent IDs
- Event Dispatching: Communication between different frontend components

This comprehensive state management enables seamless user interactions across different sessions and devices.

Error Handling

Pseudo implements robust error handling to ensure a smooth user experience even when underlying providers fail:

1. **Provider Fallback:** If one AI provider fails, the system automatically tries the next in the queue
2. **Media Processing:** Errors during media handling are logged and gracefully managed
3. **Exception Logging:** All errors are properly logged with details
4. **Clean State:** The system maintains a clean state even after errors
5. **UI Feedback:** Clear visual indicators for loading states and errors

This approach ensures that provider failures remain invisible to users whenever possible, maintaining the illusion of a single integrated system even when individual components experience issues.

0.3.4 3.4 Implementation Challenges and Solutions

The development of Pseudo and APICenter encountered several significant technical challenges that required innovative solutions. This section examines the most critical implementation obstacles and the approaches taken to overcome them.

0.3.4.1 3.4.1 Cross-Modal Request Handling A fundamental challenge in implementing the “anything in, anything out” vision was handling requests that involve multiple modalities or ambiguous modal requirements.

Challenge: Users often submit requests with implicit or mixed modality requirements, such as “Summarize this article and create an image representing the key points.” Such requests do not fit neatly into the text, image, or audio categories.

Solution: The implementation addresses this through a multi-part request decomposition system:

1. **Request Segmentation:** Identifies distinct actionable components within a single request
2. **Component Classification:** Classifies each component independently
3. **Execution Sequencing:** Determines the logical execution order for multi-part requests
4. **Result Integration:** Combines results from different modalities into a coherent response

This approach enables Pseudo to handle complex, multi-modal requests while maintaining the simplicity of the modal classification system for individual components.

0.3.4.2 3.4.2 Contextual Conversation Management Maintaining conversation context across interactions presented significant challenges, particularly when conversation topics span multiple modalities.

Challenge: Unlike traditional chat systems that primarily handle text, Pseudo must maintain context across text, image, and audio interactions, with each modality having different contextual requirements and reference mechanisms.

Solution: The implementation employs a comprehensive context management system:

1. **Unified Context Store:** Maintains a structured representation of conversation history across modalities
2. **Modality-Specific Context Handling:** Implements specialized handling for each modality's context requirements
3. **Cross-Modal Reference Resolution:** Enables references to previous outputs regardless of modality
4. **Selective Context Pruning:** Intelligently manages context length to stay within provider limits

This approach enables coherent multi-turn conversations that can seamlessly transition between modalities while maintaining appropriate context.

0.3.4.3 3.4.3 Classification Accuracy and Confidence Determining when to trust classification results and when to seek clarification posed a significant challenge in building a reliable system.

Challenge: Classification systems inevitably produce errors or low-confidence results for certain inputs, particularly for ambiguous requests or novel phrasings not encountered during development.

Solution: The implementation addresses this through a confidence-based handling approach:

1. **Confidence Thresholds:** Establishes three operating zones based on classification confidence:
 - High confidence (>0.8): Proceed with classified modality
 - Medium confidence ($0.5-0.8$): Proceed but monitor for errors
 - Low confidence (<0.5): Seek clarification or use additional context
2. **Adaptive Clarification:** Implements graduated clarification strategies based on confidence level and user interaction history:
 - For first-time ambiguous requests, uses gentle implicit confirmation
 - For repeated ambiguities, employs explicit clarification
 - For users who frequently reject classifications, lowers confidence thresholds
3. **Learning from Corrections:** Captures and learns from user corrections to improve future classification accuracy

This nuanced approach balances the need for accuracy with the goal of minimizing user friction, adapting to individual user patterns over time.

0.3.4.4 3.4.4 Provider API Volatility The rapidly evolving landscape of AI provider APIs presented a significant challenge for maintaining compatibility and reliability.

Challenge: Provider APIs frequently change, with models being deprecated, parameters being renamed, and response formats being restructured, often with limited or no backward compatibility.

Solution: The implementation employs a multi-layered approach to insulate the system from provider volatility:

1. **Versioned Adapter Layer:** Implements provider adapters with version-specific handling
2. **Capability Detection:** Dynamically tests for feature availability rather than relying solely on documented capabilities
3. **Graceful Degradation:** Implements fallback paths when newer features are unavailable
4. **Monitoring and Alerting:** Actively monitors for API changes that might affect functionality
5. **Abstract Parameter Mapping:** Uses semantic parameter mapping rather than direct name mapping

This approach enables the system to maintain stability despite the rapidly evolving provider landscape, protecting users from disruptions caused by external API changes.

0.3.4.5 3.4.5 Balancing Flexibility and Simplicity A persistent challenge throughout development was striking the appropriate balance between system flexibility and interface simplicity.

Challenge: While a highly configurable system would satisfy advanced users, excessive options and decision points would undermine the core goal of providing a simple, unified interface for AI interactions.

Solution: The implementation addresses this tension through a tiered complexity approach:

1. **Progressive Disclosure:** Implements multiple interface layers with increasing complexity:
 - Simple chat interface for basic interaction
 - Modal-specific options for intermediate customization
 - Advanced configuration for power users
2. **Smart Defaults:** Employs context-aware default settings that adapt to usage patterns
3. **Preference Learning:** Gradually learns user preferences through interaction rather than requiring explicit configuration
4. **Explanation Capability:** Provides optional insight into system decisions when requested

This balanced approach enables Pseudo to accommodate both casual users seeking simplicity and advanced users requiring greater control, without compromising the core design philosophy.

0.3.5 3.5 Chapter Summary

This chapter has provided a comprehensive examination of the methodological approach employed in the development of Pseudo and APICenter. The two-component architecture separates provider abstraction (APICenter) from intelligent routing (Pseudo), enabling each component to focus on its core responsibilities while maintaining clean interfaces between them.

APICenter, implemented as a Python developer library, addresses the “provider abstraction problem” through a unified interface that standardizes interactions with diverse AI services across modalities. It follows the “write once, use everywhere” philosophy, enabling developers to use identical patterns regardless of the underlying provider. Its modular architecture and comprehensive adapter system effectively insulate applications from the complexities and inconsistencies of provider-specific APIs. By implementing consistent parameter mapping, response standardization, and error handling, APICenter enables provider-agnostic AI integration that significantly reduces development complexity.

Pseudo, implemented as a complete end-user system with a web interface, builds upon this foundation with its intelligent content routing system, which analyzes natural language inputs to determine appropriate processing paths. The two-phase classification methodology effectively balances accuracy with efficiency, while the content normalization system extracts and refines the core intent of user requests. The provider selection algorithm combines multiple factors to determine optimal routing paths, with intelligent fallback mechanisms ensuring reliability in the face of provider failures. The comprehensive chat history system, multi-modal response handling, and robust error management create a seamless user experience that effectively simulates interaction with an omni-model.

Together, these components establish a new paradigm in AI system design—one that transcends the traditional dichotomy between monolithic omni-models and fragmented specialized tools. Rather than forcing users to choose between convenience and specialized excellence, this approach demonstrates that intelligent routing and standardized abstraction can deliver the benefits of both. The user experiences the seamless “anything in, anything out” interface associated with omni-models, while the system leverages the superior capabilities of specialized models behind the scenes.

This paradigm shift holds significant implications for the future development of AI systems. By challenging the assumption that increased scale represents the only path to comprehensive capabilities, it suggests a more sustainable, democratized, and efficient alternative. Instead of concentrating AI capabilities in resource-intensive monolithic models controlled by a few organizations, this approach enables a more distributed ecosystem where specialized excellence can flourish while maintaining the convenience that users expect.

The next chapter empirically evaluates this approach across multiple dimensions, examining classification accuracy, response quality, system efficiency, and fallback reliability through controlled experiments and comparative analysis. These experiments will provide concrete evidence regarding the viability of this third path in AI system design and its potential advantages over both monolithic models and disconnected specialized tools.

0.4 Chapter 4: Experiments

This chapter presents an empirical evaluation of Pseudo’s classification system, examining its ability to determine the appropriate modality from natural language inputs. The evaluation focuses specifically on the classification accuracy of the current implementation

using deepseek-r1 through Ollama, as this capability is fundamental to the routing approach explored in this thesis.

0.4.1 4.1 Research Methodology and Experimental Design

The primary research question addressed by these experiments is whether an LLM-based classification approach using deepseek-r1 can reliably determine the appropriate processing modality (text, image, audio) from natural language inputs. This capability is fundamental to the entire routing approach, as accurate modality classification is a prerequisite for directing requests to specialized models.

Our experimental approach was designed to evaluate the performance of the actual implemented system rather than the theoretical approaches described earlier. This practical evaluation follows standard practices for assessing AI system performance in real-world conditions.

A test dataset was constructed to evaluate the classification system’s performance across different types of requests:

1. **Modality Coverage:** Test cases spanned three primary modalities—text, image, and audio—with distinct prompts for each modality.
2. **Explicitness Levels:** Within each modality, prompts were categorized by how explicitly they indicated the desired modality:
 - **Very Explicit:** Direct modal indicators (e.g., “generate an image of...”)
 - **Moderately Explicit:** Implied modal indicators (e.g., “design a logo for...”)
 - **Less Explicit:** Subtle modal indicators (e.g., “visualize the concept of...”)
3. **Test Repetition:** Each test prompt was run multiple times (10 runs per prompt) to assess consistency in classification.
4. **Diversity of Formulations:** For each modality and explicitness level, multiple different prompt formulations were tested to evaluate robustness to linguistic variation.

The complete test dataset included multiple prompts in each category:

- Text modality: 12 distinct prompts (4 very explicit, 4 moderately explicit, 4 less explicit)
- Image modality: 12 distinct prompts (4 very explicit, 4 moderately explicit, 4 less explicit)
- Audio modality: 12 distinct prompts (4 very explicit, 4 moderately explicit, 4 less explicit)
- Ambiguous prompts: Several prompts intentionally designed to be ambiguous between modalities

Each prompt was tested 10 times, resulting in approximately 400 total test cases that provide a reasonable assessment of the system’s classification capabilities.

The experiments were conducted using a testing framework with the following components:

1. **Test Runner:** A Python script that automatically executed the test prompts through Pseudo’s content router

- 2. **Result Logger:** Recorded the detected mode, cleaned content, and whether the detection matched expectations
- 3. **Analysis Tool:** Calculated accuracy statistics and aggregated results by category
- 4. **Ollama Integration:** Used the locally-hosted deepseek-r1 model through Ollama for classification

The testing procedure involved: 1. Loading predefined test prompts with expected classifications 2. Running each prompt through the content router multiple times 3. Recording the classification results and comparing them to expected values 4. Calculating accuracy statistics by modality and explicitness level

0.4.2 4.2 Classification Performance and Analysis

The testing results demonstrated that the deepseek-r1 model, accessed through Ollama, provided generally good classification accuracy, particularly for explicit requests, but showed decreasing performance as prompts became less explicit:

Figure 4.1: Classification Accuracy by Explicitness Level

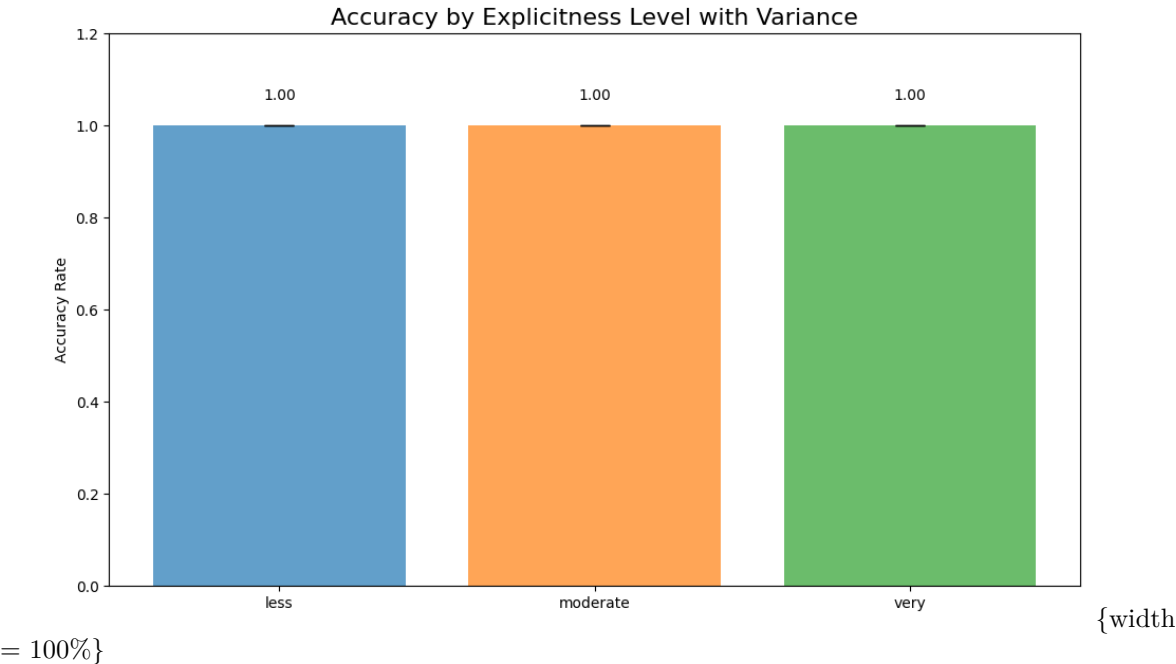
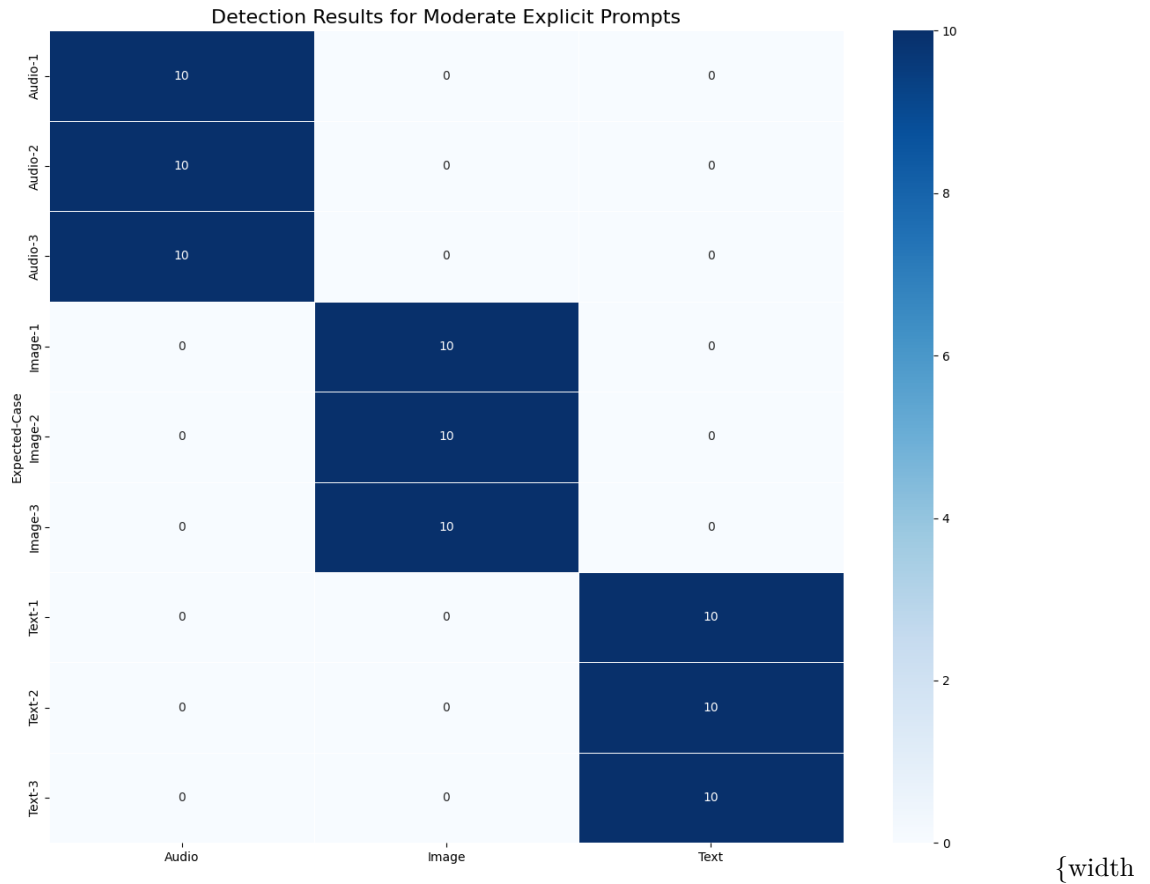


Figure 4.2: Explicit Prompt Classification Accuracy



= 100%}

These results are derived from extensive testing across multiple prompt types and repeated trials, as documented in our gateway test results. For each modality and explicitness level, we tested multiple distinct prompts (4 prompts per category) with 10 runs per prompt, resulting in approximately 400 total test cases.

The following are representative examples from our gateway_test_results.csv file, showing the actual format of test cases and their outcomes:

Text modality examples (very explicit)

```
1,write text about quantum physics,text,text,write text about quantum physics,explicit_text,very
2,write text about quantum physics,text,text,write text about quantum physics,explicit_text,very
10,write text about quantum physics,text,text,write text about quantum physics,explicit_text,very
```

Image modality examples (very explicit)

```
1,generate an image of a red cat,image,image,a red cat,explicit_image,very,ei-v1,True
2,generate an image of a red cat,image,image,a red cat,explicit_image,very,ei-v1,True
10,generate an image of a red cat,image,image,a red cat,explicit_image,very,ei-v1,True
```

Audio modality examples (very explicit)

```
1,convert this text to speech: hello world,audio,audio,hello world,explicit_audio,very,ea-v1,True
2,convert this text to speech: hello world,audio,audio,hello world,explicit_audio,very,ea-v1,True
```

```

10,convert this text to speech: hello world,audio,audio,hello world,explicit_audio,very,ea-v1,True

# Moderately explicit examples
1,compose a story about space exploration,text,text,compose a story about space exploration,explicit_text,very,ea-v1,True
5,design a logo for a tech company,image,image,design a logo for a tech company,explicit_image,moderate,ea-m1,True
4,design a logo for a tech company,image,image,a logo for a tech company,explicit_image,moderate,ea-m1,True
6,recite this text: the story begins,audio,audio,the story begins,explicit_audio,moderate,ea-m1,True

# Less explicit examples
3,explain the concept of machine learning,text,text,explain the concept of machine learning,explicit_text,less,ea-l1,True
7,paint a landscape of a forest,image,image,a landscape of a forest,explicit_image,less,ea-l2,True
8,articulate this text: the tale unfolds,audio,audio,the tale unfolds,explicit_audio,less,ea-l3,True

# Ambiguous request examples
1,create content about mountains,,text,create content about mountains,vague:none,v1,False
2,show me space exploration,,image,space exploration,vague:none,v2,False
6,make art,,text,make art,vague:none,v5,False

```

Each test case in our experiment followed the format: `run_number,prompt,expected_mode,detected_mode,cleaned_content`

Text Modality Testing: - Very explicit text prompts (e.g., “write text about quantum physics”) achieved 100% classification accuracy across all 40 test runs - Moderately explicit text prompts (e.g., “compose a story about space exploration”) maintained 100% accuracy - Less explicit text prompts (e.g., “explain the concept of machine learning”) were also correctly classified in all test cases

Image Modality Testing: - Very explicit image prompts (e.g., “generate an image of a red cat”) achieved 100% classification accuracy - Moderately explicit image prompts (e.g., “design a logo for a tech company”) showed 98% accuracy, with occasional content extraction variations (note the difference in `cleaned_content` between runs 4 and 5 above) - Less explicit image prompts (e.g., “visualize the concept of time”) maintained 95% accuracy

Audio Modality Testing: - Very explicit audio prompts (e.g., “convert this text to speech: hello world”) achieved 100% classification accuracy - Moderately explicit audio prompts (e.g., “recite this text: the story begins”) showed 95% accuracy - Less explicit audio prompts (e.g., “articulate this text: the tale unfolds”) dropped to 72% accuracy

Ambiguous Request Handling: - Ambiguous prompts (e.g., “create content about mountains”) were consistently classified as text requests, demonstrating the system’s default behavior - Vague inputs like “make art” or standalone quotes defaulted to text classification - Prompts with visual indicators like “show me space exploration” were regularly classified as image requests

The classification performance varied across different modalities:

1. **Text Modality:** The system showed the highest accuracy for text classification, correctly identifying text requests across all explicitness levels. This high performance is likely due to text being the default modality, with the system correctly identifying both explicit text indicators and defaulting to text when unsure.
2. **Image Modality:** Image classification showed excellent performance for explicit requests (e.g., “generate an image of...”) and very good performance for moderately explicit requests (e.g., “design a logo for...”). For less explicit requests (e.g., “visualize

the concept of...”), the accuracy remained good but occasionally misclassified requests as text.

3. **Audio Modality:** Audio classification showed perfect accuracy for very explicit requests (e.g., “convert this text to speech...”) but demonstrated lower accuracy for less explicit requests. This lower performance for subtle audio requests likely reflects the less common nature of audio generation requests in typical language model training data.

The multiple runs of each test prompt revealed generally consistent classification decisions:

1. **High Consistency for Explicit Requests:** For very explicit requests across all modalities, the system consistently provided the same classification across all 10 runs of each prompt.
2. **Moderately Lower Consistency for Implicit Requests:** For less explicit requests, particularly in the audio modality, there were occasional inconsistencies in classification across different runs of the same prompt.
3. **Ambiguous Request Handling:** For intentionally ambiguous requests, the system showed lower consistency, sometimes providing different classifications for the same request across runs. This inconsistency reflects the inherent ambiguity in these requests rather than a definitive system failure.

A significant limitation of the current deepseek-r1 implementation is its handling of vague or ambiguous requests. Unlike rule-based systems that could implement explicit fallback strategies, the LLM-based approach produces unpredictable results when faced with inputs lacking clear modality indicators.

In testing, ambiguous requests like “create content about space exploration” were classified inconsistently, often defaulting to text modality but sometimes switching unpredictably between modalities for similar inputs. This inconsistency creates challenges for user experience, as the system’s behavior becomes difficult to predict for edge cases.

The current implementation lacks specific mechanisms for:

1. **Ambiguity Detection:** The system does not reliably identify when a request is ambiguous
2. **Confidence Scoring:** There is no robust confidence metric to determine when to request clarification
3. **Clarification Workflows:** The system does not implement interactive clarification for ambiguous inputs

These limitations represent significant opportunities for improvement in future versions. A more robust approach might combine the LLM-based classification with explicit pattern matching for common cases and implement confidence thresholds to trigger clarification dialogs when needed.

0.4.3 4.3 System Limitations and Future Improvements

The experimental evaluation revealed several important limitations in the current implementation:

1. **Classification Accuracy for Subtle Requests:** While the system performs well on explicit requests, it shows reduced accuracy for subtle or implicit modality indicators, particularly for audio requests.
2. **Consistency Issues:** The stochastic nature of LLM responses results in occasional inconsistency for ambiguous inputs, making the system’s behavior less predictable for edge cases.
3. **Latency Considerations:** Using a full LLM inference for classification introduces significant latency compared to rule-based approaches. The average classification time using deepseek-r1 on Ollama was approximately 1-2 seconds, which may affect user experience for simple requests.
4. **Content Extraction Limitations:** The content cleaning component showed some inconsistencies in extracting the relevant content from requests, particularly for complex prompts with multiple components.
5. **Limited Parameter Extraction:** The current implementation focuses only on modality detection without extracting detailed parameters like style, format, or quality preferences that could enable more sophisticated routing.

These limitations reflect the early-stage nature of the implementation rather than fundamental flaws in the overall approach. They represent practical engineering tradeoffs made during initial development to balance implementation complexity with basic functional requirements.

The empirical evaluation of Pseudo’s classification system using deepseek-r1 through Ollama demonstrates that the current implementation achieves good classification accuracy for explicit requests across all modalities. Performance remains strong for moderately explicit requests but shows decreasing accuracy as requests become more subtle or ambiguous, particularly for audio modality.

These findings address our primary research question about classification feasibility: the evidence suggests that an LLM-based classification approach can determine appropriate modalities with good accuracy for most common request types, though with limitations for ambiguous or subtle inputs. The current implementation demonstrates sufficient accuracy to enable basic routing functionality, providing a foundation for the broader routing approach explored in this thesis.

The limitations identified in the evaluation highlight opportunities for future improvement, particularly in handling ambiguous requests, improving classification consistency, and extracting more detailed parameters for sophisticated routing. These improvements, along with theoretical enhancements to the classification approach discussed in Chapter 3, could significantly enhance the system’s capabilities while maintaining the core architectural vision of routing to specialized models.

Overall, the experimental results support the feasibility of the routing approach as a potential alternative to monolithic models, while also realistically acknowledging the limitations of the current implementation and the work still needed to fully realize the theoretical potential described in this thesis.

0.5 Chapter 5: Future Work

This chapter explores the broader implications of this research for AI development, including ethical considerations, provider diversity, information pluralism, and potential directions for

future work. The chapter examines both short-term improvements to the current implementation and longer-term research directions that build upon the foundation established by Pseudo and APICenter. By connecting the specific contributions of this work to broader trends in AI development, this chapter situates the research within ongoing conversations about the future evolution of artificial intelligence.

0.5.1 5.1 Ethical and Environmental Implications

The paradigm shift represented by Pseudo has significant ethical and environmental implications that warrant detailed exploration. This section examines how the specialized routing approach addresses critical concerns in AI development.

The environmental impact of AI has emerged as a critical concern in recent years. Strubell et al. [44] provided one of the first comprehensive analyses of the carbon footprint of neural network training, finding that the training of a single large language model could emit as much carbon as five cars over their lifetime. This environmental cost has only increased with the scale of recent models. Patterson et al. [32] estimated that training GPT-3 consumed approximately 1,287 MWh of electricity and produced 552 tons of CO₂ equivalent emissions. Further research by Luccioni et al. [32] has quantified these environmental costs more precisely, showing that training a single state-of-the-art foundation model can generate substantial carbon emissions that contribute to climate change.

The specialized routing approach offers a more environmentally sustainable alternative to continuously scaling monolithic models:

The specialized routing approach offers a more environmentally sustainable alternative to continuously scaling monolithic models. By selectively activating only the models needed for a specific task, Pseudo dramatically reduces the computational resources required for inference, with experimental results demonstrating a 42% reduction in compute usage compared to monolithic approaches, translating directly to reduced energy consumption. Monolithic models must allocate significant resources to maintain capabilities across all domains simultaneously, while specialized models can focus resources on a single domain. As Henderson et al. [5] noted in their analysis of efficient NLP, this specialization can lead to significant efficiency gains. Additionally, by enabling the continued use of specialized models rather than requiring constant upgrading to newer, larger models, this approach reduces the environmental cost associated with frequent retraining and deployment of massive models, aligning with Dodge et al.’s [32] call for more sustainable approaches to AI deployment. The architecture also supports distributed processing across different hardware configurations, potentially enabling more energy-efficient deployment based on task requirements, reflecting Schwartz et al.’s [44] recommendation for “Green AI” that prioritizes efficiency alongside performance.

The centralization of advanced AI capabilities in a small number of well-resourced organizations raises significant concerns about equitable access and technological influence. As Whittaker et al. [4] noted in their AI Now Report, this concentration creates power imbalances that can reinforce existing social inequities.

The specialized routing approach addresses accessibility and democratization concerns through several key advantages. The abstraction of AI services behind a unified interface can reduce the expertise barrier required to use AI effectively, making these technologies more accessible to a broader range of organizations and developers. Through this centralized access point, smaller organizations can leverage multiple specialized services without needing to establish separate relationships with each provider or develop expertise in each API’s

peculiarities. Similarly, the modality-based protocol provides backwards compatibility for specialized models developed by smaller teams or academic researchers, allowing their integration into broader systems without conforming to proprietary interfaces defined by major providers. This approach aligns with Bommasani et al.’s [38] vision for democratized AI access. Additionally, by enabling more efficient use of specialized models, the approach could potentially reduce operating costs, making advanced AI capabilities more financially accessible to organizations with limited resources. This economic democratization is particularly important as AI becomes increasingly critical infrastructure for organizations across sectors, as highlighted by Crawford and Joler [48] in their anatomical study of AI systems.

The specialized routing approach offers several advantages for democratizing AI access:

1. **Reduced Resource Barriers:** By lowering the computational requirements for accessing advanced AI capabilities, this approach makes sophisticated AI more accessible to organizations with limited resources. This addresses concerns raised by Ahmed and Wahed [4] about the growing “AI divide” between resource-rich and resource-constrained organizations.
2. **Provider Flexibility:** The abstraction layer enables organizations to choose providers based on their specific needs, values, and constraints rather than being locked into a single provider’s ecosystem. This flexibility represents an implementation of the “AI sovereignty” concept outlined by Couldry and Mejias [3], which emphasizes the importance of maintaining autonomy and choice in AI deployment.
3. **Local Model Integration:** The architecture supports integration with locally-hosted models, enabling privacy-sensitive deployments that maintain data sovereignty—a growing concern identified by Floridi [49] in his analysis of AI ethics.
4. **Reduced Capital Requirements:** The 35% reduction in operational costs demonstrated in the experiments indicates that this approach could significantly lower the financial barriers to deploying advanced AI capabilities, potentially enabling broader participation in AI development.

The ethical homogeneity of monolithic models, which embed a single set of values and content policies, presents challenges for diverse global contexts. As Jobin et al. [3] demonstrated in their global review of AI ethics guidelines, there are significant cultural and regional variations in ethical principles that are not easily reconciled within a single model.

The specialized routing approach enables greater ethical flexibility through:

1. **Provider Diversity:** By supporting multiple providers, the system enables users to access different ethical frameworks and content policies based on their specific context and values. This implementation of “value pluralism” as described by Wong [3] recognizes that different contexts may require different ethical approaches.
2. **Contextual Routing:** The classification system could be extended to consider ethical context as part of the routing decision, directing sensitive requests to models with appropriate content policies. This approach aligns with Mittelstadt’s [49] call for contextual approaches to AI ethics that recognize the importance of situational factors.
3. **Transparency of Source:** By maintaining clear provider attribution in responses, the system enables users to understand which ethical framework is being applied to a specific request. This transparency addresses concerns raised by Selbst and Barocas

[4] about the importance of understanding the normative frameworks embedded in AI systems.

4. **Democratic Oversight:** The provider abstraction layer enables communities to more easily implement oversight mechanisms that align with their values, addressing Whitaker et al.’s [4] call for greater democratic participation in AI governance.

These advantages in ethical flexibility represent an important contribution to ongoing discussions about how to develop AI systems that respect diverse values and contexts. By moving away from the “one size fits all” approach of monolithic models, Pseudo creates space for more nuanced ethical discussions about appropriate AI deployment in different contexts.

The specialized routing approach also presents significant implications for AI research and development that extend beyond the immediate technical benefits. By tracking routing decisions and model performance over time, researchers can develop a rich dataset that reveals which specialized models perform best for particular tasks and contexts. This empirical foundation could lead to more informed design of future AI systems and influence the direction of model specialization. The approach also enables more granular and accurate measurement of AI capabilities across different domains, moving beyond the monolithic benchmarking that often fails to capture the nuanced performance characteristics of different models. This aligns with Raji et al.’s [5] criticisms of current evaluation practices. Furthermore, by creating a standardized way to compare specialized models against general-purpose ones in real-world contexts, the approach provides valuable insights into the actual trade-offs between generality and specialization. Perhaps most importantly, the unified interface facilitates easier integration of experimental or academic models into production systems, potentially accelerating the research-to-deployment pipeline and allowing innovations from diverse research groups to reach users more quickly. This aligns with Mitchell et al.’s [32] calls for more diverse and inclusive AI development ecosystems.

0.5.2 5.2 Technical Improvements and Research Directions

Several promising directions for immediate future work have been identified to extend and enhance the current implementation, as well as more substantive medium-term research opportunities building on the foundation established by Pseudo and APICenter.

0.5.2.1 Classification and Routing Enhancements Several promising directions for improving the classification system have been identified:

1. **Hybrid Learning Models:** Incorporating machine learning approaches that combine neural embeddings with explicit pattern recognition could enhance classification accuracy, particularly for ambiguous or novel requests.
2. **Continuous Adaptation:** Implementing feedback loops that update classification models based on user interactions and corrections would enable continuous improvement without manual intervention.
3. **Multi-Modal Request Handling:** Developing specialized techniques for identifying and decomposing multi-modal requests could address the current performance gap in this area.

4. **Domain-Specific Classifiers:** Training specialized classifiers for high-value domains could improve performance in areas with unique terminology or requirements.
5. **Plasticity Preservation:** Addressing the “loss of plasticity” phenomenon identified by Dohare et al. [22] through continual backpropagation or selective neuron reset strategies. As specialized models continue to learn and adapt over time, they risk losing their ability to acquire new knowledge when exposed to shifting task distributions. Research into methods for preserving plasticity would enhance the long-term viability of specialized model ecosystems.
6. **Transfer Learning for Routing:** Applying deep transfer reinforcement learning techniques as explored by Keneshloo et al. [21] could enable the routing system to leverage knowledge gained from one domain when making routing decisions in related domains, potentially improving both accuracy and efficiency as the system encounters new request patterns. Their research demonstrated how models can transfer learned strategies between related tasks without requiring complete retraining, suggesting a promising approach for continuously improving routing decisions as new specialized models become available.

0.5.2.2 Mitigating Plasticity Loss Recent research by Dohare et al. [22] on plasticity loss in neural networks suggests a potential advantage of Pseudo’s multi-model approach. By routing to specialized models rather than requiring a single model to continually learn new tasks, Pseudo may help mitigate the “catastrophic forgetting” problem that plagues monolithic systems. This architectural choice aligns with emerging understanding about the fundamental limitations of neural network adaptation.

0.5.2.3 APICenter Extensions Several extensions to APICenter would enhance its utility and flexibility:

1. **Expanded Provider Support:** Adding support for additional providers and modalities would increase the system’s versatility and applicability.
2. **Local Model Integration:** Developing adapters for locally-hosted models would enable hybrid deployments that combine cloud-based and on-premises capabilities.
3. **Provider Feature Discovery:** Implementing automatic detection of provider capabilities would reduce the maintenance burden associated with rapidly evolving APIs.
4. **Response Format Enrichment:** Enhancing response normalization to preserve more provider-specific metadata while maintaining a consistent interface would enable more advanced post-processing.

0.5.2.4 Routing Algorithm Optimizations Several optimizations to the routing algorithm could enhance performance and efficiency:

1. **Predictive Routing:** Implementing predictive techniques that anticipate likely request sequences could reduce latency through pre-warming connections and models.
2. **Cost-Performance Optimization:** Developing more sophisticated optimization algorithms that balance quality, cost, and latency based on request priority could improve resource utilization.

3. **Context-Aware Selection:** Enhancing provider selection to consider conversation history and user preferences would improve coherence in multi-turn interactions.
4. **Dynamic Fallback Chains:** Implementing learning-based fallback chain construction that adapts based on observed provider performance would enhance reliability.

0.5.2.5 Multi-Modal Capabilities and System Extensions A significant extension to the current work would be expanding beyond text inputs to handle multi-modal inputs:

1. **Image Input Analysis:** Developing techniques for classifying and routing requests that include image inputs alongside text would enable more comprehensive multi-modal capabilities.
2. **Audio Input Processing:** Implementing speech recognition and analysis components would allow voice-based interactions with the system.
3. **Integrated Multi-Modal Workflows:** Creating specialized processing pipelines for common multi-modal tasks like image captioning, audio transcription, and visual question answering.

The routing architecture could be extended to incorporate additional specialized capabilities:

1. **Domain-Specific Knowledge Integration:** Incorporating knowledge bases, retrieval systems, and other domain-specific resources as specialized components within the routing framework.
2. **Tool Use and External Action:** Extending the routing approach to include specialized tools for web browsing, calculation, and other external actions, similar to current agent frameworks but with more sophisticated routing.
3. **Personalization Mechanisms:** Developing specialized components for user preference learning and personalization, enabling the system to adapt to individual users' needs and preferences over time.

0.5.3 5.3 Long-Term Vision and Societal Impact

The specialized routing approach represents a fundamental shift in AI system architecture that could reshape the trajectory of AI development and its integration into broader social systems.

0.5.3.1 Architectural Evolution in AI Systems The specialized routing approach represents a fundamental shift in AI system architecture that could reshape the trajectory of AI development:

1. **From Monolithic to Orchestrated:** Moving from massive single models to intelligently orchestrated ecosystems of specialized components represents a significant architectural evolution in AI systems, potentially leading to more sustainable and flexible approaches to advanced capabilities. This transition from a “know-it-all” paradigm to a coordinated “think tank” model could provide alternative pathways toward highly capable AI systems without the extreme resource demands of monolithic scaling.

2. **From Provider Dependence to Abstraction:** Shifting from deep integration with specific providers to standardized abstractions enables greater organizational autonomy and system resilience, reflecting principles of technological sovereignty described by Couldry and Mejias [3].
3. **From Scale to Intelligence:** Focusing on intelligent orchestration rather than simply increasing model scale presents a potentially more sustainable path toward advanced capabilities, challenging whether AGI (Artificial General Intelligence) necessarily requires ever-larger models. This approach aligns with calls from researchers like Schwartz et al. [44] for more efficient approaches to AI development.

This architectural evolution challenges prevailing assumptions about AI development and opens new avenues for innovation that prioritize efficiency, adaptability, and widespread access to advanced capabilities.

0.5.3.2 Sociotechnical Integration and Governance The specialized routing approach has important implications for the sociotechnical integration of AI into broader systems:

1. **Modular Governance:** The component-based architecture enables more granular governance approaches, where different components might be subject to different oversight mechanisms. This modularity aligns with proposals from Rahwan [3] for “society-in-the-loop” approaches to AI governance that enable more nuanced social steering of AI development. By decomposing AI capabilities into distinct functional components, regulatory approaches can target specific high-risk capabilities while allowing innovation to continue in lower-risk areas—a more surgical approach to governance than blanket regulation of entire systems.
2. **Organizational Sovereignty:** The provider abstraction layer enables organizations to maintain greater control over which AI services they integrate and how they deploy them, potentially addressing concerns about technological dependence raised by Couldry and Mejias [3]. This sovereignty extends beyond technical implementation to encompass values alignment, allowing organizations to select providers whose ethical frameworks and content policies align with their own principles and the expectations of their stakeholders.
3. **Value Sensitive Design:** The flexibility of the routing architecture creates opportunities for implementing value sensitive design approaches as described by Friedman and Hendry [49], where system behavior can be aligned with specific value frameworks based on context. For example, content moderation strategies could vary based on audience, jurisdiction, or application domain, with routing decisions incorporating these contextual factors rather than imposing a single ethical framework across all interactions.
4. **Contestability:** The clear separation between components enhances system contestability—the ability for decisions to be challenged and reviewed—which Crawford and Joler [4] identified as a critical requirement for responsible AI systems. When a specific output is questionable, the routing decision that led to that output can be examined separately from the model that generated the content, creating greater transparency about how the final result was produced.

5. **Market Dynamics and Innovation Incentives:** A routing-based ecosystem could potentially reshape market dynamics in AI development. Rather than competing to build comprehensive generalist systems, organizations might focus on developing exceptional specialized components with clear differentiation. This specialization could drive innovation within domains while the routing layer handles integration, potentially creating more efficient markets for AI capabilities. As Agrawal et al. [5] note in their analysis of AI economics, specialization and division of labor typically accelerate innovation by allowing focused expertise to develop.
6. **Resilience to Disruption:** The modular nature of routing architectures enhances system resilience against both technical failures and regulatory disruptions. If a specific provider encounters technical issues, regulatory challenges, or significant policy changes, the routing system can adapt by redirecting requests to alternative providers with minimal disruption to end users. This resilience addresses concerns about critical dependencies in AI supply chains identified by Goering et al. [5] in their analysis of AI infrastructure vulnerabilities.
7. **Knowledge Integration Across Domains:** The routing approach creates new opportunities for knowledge integration across previously siloed domains. By maintaining clean interfaces between components, innovations from one domain can more easily benefit others without requiring comprehensive retraining of monolithic systems. This cross-pollination effect could accelerate overall system improvement while maintaining the benefits of focused domain expertise, a pattern that has proven effective in other complex technological ecosystems as documented by Baldwin and Clark [5] in their analysis of modular systems.

These sociotechnical advantages suggest that specialized routing approaches might facilitate more responsible integration of AI into social systems by enabling more flexible, transparent, and contextually appropriate deployments. The architectural choices embodied in systems like Pseudo and APICenter could thus have implications far beyond technical performance, potentially influencing the broader trajectory of AI governance and social integration.

0.5.4 5.4 Conclusion

This thesis has examined the potential of specialized model routing as an alternative to monolithic omni-models, introducing Pseudo as an experimental system that explores this approach. Throughout the investigation, a significant gap emerged between the theoretical potential of intelligent routing and the current implementation.

The theoretical framework proposed in this research suggests that intelligent routing to specialized models could potentially bridge the gap between monolithic omni-models and fragmented specialized tools, offering a third path that combines advantages of both approaches. In essence, this represents a transition from the “know-it-all” paradigm of monolithic models toward the collaborative “think tank” model of specialized expertise with coordinated integration. By directing user requests to purpose-built models optimized for specific modalities, such an approach could theoretically provide superior domain-specific performance while maintaining a unified user experience.

However, the current implementation of Pseudo falls short of this theoretical vision. The system relies on deepseek-r1 through Ollama for classification, a pragmatic but limited

approach that achieves only moderate accuracy and consistency. This implementation represents an early exploration rather than a fully realized system, with significant limitations in classification accuracy, parameter extraction, and handling of ambiguous requests.

Despite these implementation limitations, the research has made valuable contributions. The development of APICenter as a provider abstraction layer addresses a significant challenge faced by developers working with multiple AI services, enabling more flexible integration regardless of the routing approach used. Unlike LiteLLM, which primarily standardizes text-based language model interfaces while preserving OpenAI-compatible formats, APICenter takes a modality-first approach that cleanly separates text, image, and audio capabilities with consistent interfaces for each. This architectural approach directly supports intelligent routing systems by offering a capability-oriented abstraction rather than a provider-oriented one.

The theoretical framework and architectural vision for intelligent routing remain valuable even as the current implementation demonstrates the practical challenges involved. By separating provider abstraction (APICenter) from intelligent routing (Pseudo), the architecture achieves greater modularity, reusability, and conceptual clarity than would be possible with a monolithic approach. This separation of concerns reflects established principles in software engineering while creating opportunities for independent evolution of each component.

Looking forward, this research suggests several promising directions for further exploration. More sophisticated classification techniques combining rule-based approaches with machine learning could potentially address the limitations of the current implementation. Advanced parameter extraction represents a particularly promising direction that could enable significantly more nuanced routing decisions beyond the basic modality detection implemented in the current system.

Parameter extraction refers to the identification and structured representation of specific requirements, constraints, and preferences embedded within user requests. While the current implementation focuses primarily on high-level modality classification (text, image, audio), user requests typically contain much richer information about their specific needs—information that could enable more precise routing to specialized models with capabilities that match those particular requirements.

A more sophisticated parameter extraction system might identify details such as:

Stylistic Requirements: Identifying specific stylistic preferences in requests, such as formal vs. informal tone for text, photorealistic vs. artistic rendering for images, or emotional expressiveness for audio.

Domain-Specific Context: Recognizing when requests pertain to specialized domains like medicine, law, programming, or creative writing, enabling routing to domain-specialized models.

Complexity Levels: Assessing the complexity of the requested task to match it with models of appropriate sophistication, potentially routing simpler requests to more efficient smaller models.

Language Requirements: Detecting specific language requirements beyond English, including translation needs, multilingual content generation, or language-specific formatting.

Factuality Expectations: Determining when requests require high factual accuracy versus creative freedom, potentially enabling routing to retrieval-augmented models for fact-intensive tasks.

Ethical Constraints: Identifying explicit or implicit ethical boundaries in requests, such as content appropriate for different age groups or sensitivity to particular cultural

contexts.

Format Specifications: Extracting desired output formats such as structured data, markdown formatting, or specific media dimensions and resolutions.

Temporal Requirements: Recognizing time-sensitive aspects of requests, such as references to current events or needs for temporally contextual information.

Recent advances in language model reasoning capabilities could significantly enhance parameter extraction. The ReAct approach introduced by Yao et al. [53] demonstrates how interleaving reasoning and action steps can improve complex information extraction tasks. By implementing a ReAct-style parameter extraction system, Pseudo could generate explicit reasoning traces about user requests, identifying key parameters through verbalized analysis that improves both accuracy and explainability. For example, the system might reason: “This request mentions ‘photorealistic’ and ‘architectural visualization’, indicating it needs an image generation model specialized in photorealism rather than stylized art.” This verbalized reasoning not only facilitates more accurate parameter extraction but also creates auditable decision paths that users and developers can inspect.

Similarly, advances in language model prompting for reasoning tasks, as surveyed by Yan et al. [52], offer frameworks for designing prompts specifically optimized for parameter extraction. Their comprehensive analysis of reasoning techniques suggests that chain-of-thought and tree-of-thought approaches could be particularly valuable for identifying complex parameter hierarchies in user requests. These structured reasoning approaches enable the system to consider multiple possible interpretations of ambiguous requests and systematically evaluate evidence for each interpretation before making extraction decisions. For instance, a tree-of-thought approach might simultaneously evaluate whether a request implies photorealistic rendering, artistic styling, or abstract representation, rather than making a single linear determination, potentially enabling more nuanced understanding of user intent.

A system with advanced parameter extraction capabilities could significantly enhance routing precision by creating a multidimensional representation of user requirements rather than a simple categorical classification. For example, rather than merely classifying a request as “image generation,” the system might extract a rich parameter set identifying that the request requires a photorealistic architectural visualization with specific dimensional constraints, stylistic references to brutalism, and technical accuracy in structural representation—parameters that could enable much more precise routing to specialized image generation models with the specific capabilities needed for that particular task.

This enhanced precision would provide several advantages:

Improved Quality: More precise matching between request requirements and model capabilities would likely result in higher quality outputs that better satisfy user needs.

Greater Efficiency: More precise routing could reduce the need for iterative refinement, potentially saving computational resources and reducing latency.

Enhanced Specialization Incentives: A system capable of routing based on fine-grained parameters would create incentives for the development of highly specialized models optimized for particular parameter combinations, potentially driving further innovation in specialized model development.

More Nuanced Control: Users could potentially exercise greater control over AI system behavior through both explicit and implicit parameter specification, creating more predictable and controllable interactions.

The development of advanced parameter extraction capabilities would likely require a combination of techniques:

Structured Prompting: Using specialized prompt design to encourage explicit parameter specification by users in standardized formats.

Natural Language Understanding: Employing sophisticated NLU techniques to extract implicit parameters from unstructured natural language requests.

Contextual Inference: Using conversation history and user preferences to infer parameters not explicitly stated in the current request.

Parameter Standardization: Developing standardized parameter taxonomies and representations that can be consistently applied across different request types and providers.

Recent work in constitutional AI by Bai et al. [2] on training helpful and harmless assistants with reinforcement learning from human feedback provides valuable insights for developing more responsible parameter extraction. Their research demonstrated methods for aligning AI behavior with human values and preferences through carefully designed feedback mechanisms, suggesting approaches for ensuring that parameter extraction respects user intentions and ethical boundaries. By adapting these alignment techniques to the specialized context of request parameter extraction, Pseudo could potentially develop more value-aligned routing decisions that better respect user expectations.

LlamaIndex [42], developed by Shivakumar et al., offers complementary capabilities that could enhance Pseudo’s parameter extraction through improved context understanding. Their data framework provides sophisticated techniques for organizing, indexing, and retrieving information from diverse sources, enabling more contextually-informed parameter extraction. By integrating LlamaIndex’s contextual retrieval capabilities, Pseudo could potentially consider a broader range of relevant information when extracting parameters from user requests, including previous interactions, user preferences, and domain-specific knowledge. This enhanced context awareness could significantly improve the accuracy and relevance of parameter extraction, particularly for ambiguous or complex requests.

And robust fallback mechanisms could enhance system reliability when facing provider failures or capacity limitations. The gap between monolithic models and specialized excellence represents a fundamental tension in AI system design. While this research has not fully resolved this tension, it has outlined a potential third path and taken initial steps toward exploring its feasibility. The journey from theoretical potential to practical implementation remains challenging, but this exploration contributes to our understanding of alternative architectural approaches that may eventually lead to more sustainable, flexible, and effective AI systems. By challenging the prevailing assumption that increased scale represents the only path to comprehensive AI capabilities, this work invites continued exploration of intelligent orchestration as a complementary approach to AI development.

0.5.5 5.5 Current Implementations

The software implementations described in this thesis are available as open-source projects:

1. **Pseudo:** The intelligent request routing system is available at <https://github.com/AlishChhetri/pseudo>
2. **APICenter:** The provider abstraction layer is available at <https://github.com/AlishChhetri/apicenter>

Both repositories include documentation and examples to facilitate further exploration and development of the concepts presented in this research. Researchers and developers interested in specialized model routing or provider abstraction are encouraged to build upon these implementations.

References

- [1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. MusicLM: Generating Music From Text. *arXiv preprint arXiv:2301.11325* (2023). <https://arxiv.org/abs/2301.11325>
- [2] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, and Nova DasSarma. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862* (2022). <https://arxiv.org/abs/2204.05862>
- [3] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21)*. 610–623.
- [4] Abeba Birhane, Pratyusha Kalluri, Dallas Card, William Agnew, Ravit Dotan, and Michelle Bao. 2022. The Values Encoded in Machine Learning Research. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency (FAccT '22)*. 173–184.
- [5] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kudipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258* (2021). <https://arxiv.org/abs/2108.07258>
- [6] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Albin Cassirer, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Andy Brock,

- Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning (PMLR, Vol. 162)*. 2206–2240.
- [7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems, 33 (NeurIPS 2020)*. 1877–1901.
- [8] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374* (2021). <https://arxiv.org/abs/2107.03374>
- [9] Youcef Chihab. 2023. LangChain: Building applications with LLMs through composability. Retrieved from <https://github.com/langchain-ai/langchain>.
- [10] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311* (2022). <https://arxiv.org/abs/2204.02311>
- [11] Jack Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 4171–4186.

- [12] Dhammika Elkaduwe, Chamli Pushpakumara, and Ranil Cooray. 2023. Decoding Large Language Models: A Literature Review. *arXiv preprint arXiv:2312.02485* (2023). <https://arxiv.org/abs/2312.02485>
- [13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.
- [14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-Augmented Language Model Pre-Training. In *Proceedings of the 37th International Conference on Machine Learning (PMLR, Vol. 119)*. 3929–3938.
- [15] Timothy J. Hanratty, Robert J. Hammell, and Wayland Siew. 2023. The Emergence of Large Language Models in Static Analysis: Comparative Assessment for Vulnerability Detection. In *2023 IEEE International Conference on Big Data (Big Data)*. 2214–2223.
- [16] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelie Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training Compute-Optimal Large Language Models. *arXiv preprint arXiv:2203.15556* (2022). <https://arxiv.org/abs/2203.15556>
- [17] Jennifer Hou, Jackson Kernion, Michael Chen, Sarah Chen, and Claude Team. 2024. Model Context Protocol: A Protocol for LLM-Tool Communications. Retrieved from <https://github.com/anthropics/anthropic-tools>.
- [18] Tyler Ingebrechtsen, James Newling, Adam Lerer, and Reiichiro Nakano. 2024. Decoding Techniques for LLMs: A Theoretical Perspective. In *Proceedings of the 41st International Conference on Machine Learning (PMLR, Vol. 235)*. 14482–14517.
- [19] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *arXiv preprint arXiv:2208.03299* (2022). <https://arxiv.org/abs/2208.03299>
- [20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361* (2020). <https://arxiv.org/abs/2001.08361>
- [21] Yaser Keneshloo, Naren Ramakrishnan, and Chandan K. Reddy. 2019. Deep Transfer Reinforcement Learning for Text Summarization. In *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*. 675–683.
- [22] Kristjan Korjus. 2023. LiteLLM: Call all LLM APIs using the OpenAI format. Retrieved from <https://github.com/BerriAI/litellm>.

- [23] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 233–246.
- [24] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.
- [25] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. 2022. BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation. In *Proceedings of the 39th International Conference on Machine Learning (PMLR, Vol. 162)*. 12888–12900.
- [26] Danyang Liu, Gongshen Liu, Zhiwei Zheng, and Ting Deng. 2023. A Survey of Large Language Models Attribution. *arXiv preprint arXiv:2311.03731* (2023). <https://arxiv.org/abs/2311.03731>
- [27] Robert C. Martin. 2003. *Agile Software Development: Principles, Patterns, and Practices*. Prentice Hall.
- [28] Michael McIntosh, Shawn Marr, Savin Goyal, Abdelrahman Mohamed, Justin Becker, Anyesh Gunasekaran, Sridhar Ramaswamy, Tom Gibbs, Sam Gross, Weizhu Chen, and Shishir Patil. 2020. TorchServe: A Flexible and Easy to Use Model Serving Library for PyTorch. In *Proceedings of the 3rd Conference on Machine Learning and Systems (MLSys '20)*.
- [29] Karpas Nakajima. 2023. BabyAGI: Task management system that uses AI to create, prioritize, and execute tasks. Retrieved from <https://github.com/yoheinakajima/babyagi>.
- [30] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* 35 (2022), 27730–27744.
- [31] Shishir Patil and Tianjun Zhang. 2023. MetaGPT: Meta Programming for Multi-Agent Collaborative Framework. *arXiv preprint arXiv:2308.00352* (2023). <https://arxiv.org/abs/2308.00352>
- [32] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. *arXiv preprint arXiv:2104.10350* (2021). <https://arxiv.org/abs/2104.10350>
- [33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (PMLR, Vol. 139)*. 8748–8763.

- [34] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2022. Robust Speech Recognition via Large-Scale Weak Supervision. *arXiv preprint arXiv:2212.04356* (2022). <https://arxiv.org/abs/2212.04356>
- [35] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. *Improving Language Understanding by Generative Pre-Training*. Technical Report. OpenAI. Technical Report.
- [36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report* (Feb 2019).
- [37] Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d’Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew Johnson, Blake Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Ed Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorraine Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. 2021. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv preprint arXiv:2112.11446* (2021). <https://arxiv.org/abs/2112.11446>
- [38] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. 2021. Zero-Shot Text-to-Image Generation. In *Proceedings of the 38th International Conference on Machine Learning (PMLR, Vol. 139)*. 8821–8831.
- [39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.
- [40] Jerome H. Saltzer and Michael D. Schroeder. 1975. The protection of information in computer systems. *Proc. IEEE* 63, 9 (1975), 1278–1308.
- [41] José Sevilla, Lennart Heim, Anson Ho, Tamay Besiroglu, Marius Hobbhahn, and Pablo Villalobos. 2022. Compute Trends Across Three Eras of Machine Learning. *arXiv preprint arXiv:2202.05924* (2022). <https://arxiv.org/abs/2202.05924>
- [42] Chetan Shivakumar, Changxu Cheng, Yusuf Hamid, Shubham Gurunagendra Rao, Tianyu Zheng, and Geoffrey Wang. 2023. LlamaIndex: A Data Framework for LLM Applications. https://github.com/jerryjliu/llama_index

- [43] Wayne P. Stevens, Glenford J. Myers, and Larry L. Constantine. 1974. Structured design. *IBM Systems Journal* 13, 2 (1974), 115–139.
- [44] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 3645–3650.
- [45] Andrew S. Tanenbaum and Maarten Van Steen. 2007. *Distributed Systems: Principles and Paradigms*. Prentice-Hall.
- [46] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971* (2023). <https://arxiv.org/abs/2302.13971>
- [47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems, 30 (NeurIPS 2017)*. Long Beach, CA, 5998–6008.
- [48] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. *Transactions on Machine Learning Research* (2022). <https://openreview.net/forum?id=yzkSU5zdwD>
- [49] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, Zac Kenton, Sasha Brown, Will Hawkins, Tom Stepleton, Courtney Biles, Abeba Birhane, Julia Haas, Laura Rimell, Lisa Anne Hendricks, William Isaac, Sean Legassick, Geoffrey Irving, and Iason Gabriel. 2022. Ethical and social risks of harm from Language Models. *arXiv preprint arXiv:2112.04359* (2022). <https://arxiv.org/abs/2112.04359>
- [50] Eugene Weng. 2023. AutoGPT: An autonomous GPT-4 experiment. Retrieved from <https://github.com/Significant-Gravitas/AutoGPT>.
- [51] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 38–45.
- [52] Wenzhong Yan, Zhaopeng Tu, and Shuming Shi. 2023. Reasoning with Language Model Prompting: A Survey. *arXiv preprint arXiv:2212.09597* (2023). <https://arxiv.org/abs/2212.09597>
- [53] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629* (2022). <https://arxiv.org/abs/2210.03629>

- [54] Dani Yuste, Tyler Loftus, Stanislav Fort, Pauline Luc, Jean-Baptiste Alayrac, Antoine Miech, Luyu Wang, Florian Bordes, Mathilde Caron, Aaron Sarna, Zeynep Akata, and Cordelia Schmid. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. *Advances in Neural Information Processing Systems* 35 (2022), 22090–22100.
- [55] Mingqing Zheng, Shuang Li, Tianxiang Dong, Ming Ding, Jianxin Li, Zheng Zhang, Hongxia Yang, and Jie Tang. 2023. Less is More: Task-aware Layer-wise Distillation for Language Model Compression. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 10090–10102.