Modelling Cognitive Brain Processes

CS 298 Report

Presented To

The Department of Computer Science

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science in Computer Science

By

Adithya Harish Kumar

December, 2025

The Designated Project Committee Approves the Project Titled

Modelling Cognitive Brain Processes

By

Adithya Harish Kumar

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2025

Dr. Chris Pollett, Department of Computer Science

Dr. Robert Chun, Department of Computer Science

Dr. Sayma Akther, Department of Computer Science

ABSTRACT

Agent based systems can simulate a variety of complex environments, social behavior and cognitive processes. Since the heart of these agents, Large Language Models (LLMs) are trained on data generated by humans, they offer a route to seamless human cognition modelling. In this work, we build simulations of the visual cortex using the dorsal-ventral model of cognitive vision and further extend it to model V1–V6 visual processing subsystems, that perform functions like contour detection, binocular disparity, motion perception and color perception that support visual processing in the human brain. To implement this, we developed a framework, Topology Manger to enable deployment and management of LLM agents with customizable interaction patterns, memory management, and support for various LLM backends. We also finetuned a Phi-3.5-mini-instruct model on the ReClor logical reasoning dataset, achieving a prediction accuracy of 89%, which will later be used to maintain coherent reasoning across the simulated brain regions. Using these components, we construct a system that predicts human visual attention given an input image, achieving a precision score of 0.6 on a standard visual saliency dataset. This work demonstrates the potential of multi-agent LLM systems in modeling brain processes and contributes toward building cognitively grounded AI architectures.

Keywords – Visual Cortex, AI Agents, Topology Manager, Phi-3.5-mini-instruct

ACKNOWLEGEMENT

I would like to sincerely thank my project advisor, Dr. Chris Pollett, for his invaluable guidance, support, and encouragement throughout the course of this work. Our numerous discussions were instrumental in shaping the direction of this project, helping to refine ideas, explore new perspectives, and overcome critical challenges. His mentorship not only enhanced the technical depth of the project but also greatly contributed to my overall growth as a researcher. I am also grateful to Dr. Robert Chun and Dr. Sayma Akhter for their valuable feedback, insightful suggestions, and critical evaluation, which greatly helped in improving the quality of this project. Finally, I would like to thank the Department of Computer Science at San José State University for providing the necessary resources and academic environment that made this project possible.

TABLE OF CONTENTS

1. INTRODUCTION

Large Language Models (LLMs) can simulate multi-stage cognitive and perceptual processes by representing each stage as a modular task performed by specialized agents. By organizing these agents into structured topologies, LLMs can mimic biologically inspired workflows, maintain context across stages, and adapt to multimodal inputs. This approach enables the construction of AI systems capable of handling complex reasoning and perception tasks in a modular and interpretable manner.

Models of consciousness aim to explain the mechanisms underlying conscious thought and perception and can be implemented using Large Language Models (LLMs) to simulate these processes. Dennett's Multiple Drafts model suggests that consciousness arises from parallel processes in the brain that create multiple competing narratives [2], with no central observer deciding which is the "true" perception, a concept that can be mirrored by LLMs operating in distributed, narrative-focused roles. The Dehaene-Changeux model, which proposes a Global Neuronal Workspace for broadcasting and integrating information [1], can be simulated by orchestrating LLMs in a hierarchical architecture where key outputs are shared across interconnected agents. These implementations provide computational tools to explore and validate theories of consciousness.

Models of human visual perception, particularly those describing the visual cortex, can be computationally approximated using LLMs arranged in topologies that mirror biological structures. In this research, agents were choreographed to simulate regions such as V2 and V4. The V2 agent was designed to process edge detection, spatial frequency, and contour integration, while the V4 agent was responsible for object recognition, shape analysis, and color perception. Outputs from these agents were used as contextual inputs for a fine-tuned reasoning model based on Phi-3.5-mini-instruct, which performed final decision-making and query resolution.

Agentic Retrieval-Augmented Generation (RAG) enables LLM-based agents to query external data sources and combine retrieved information with generative reasoning. Existing frameworks such as CrewAI demonstrate this by coordinating multiple agents around distinct subtasks. This research extends that principle by designing and implementing the LLM Topology Manager - a framework that allows users to declaratively configure the structure and interaction pathways of LLM agents, supporting multimodal inputs and enabling systematic exploration of cognitive architectures.

To evaluate the capabilities of these agentic systems in handling multimodal tasks, the Multimodal Massive Multi-Discipline Understanding (MMMU) benchmark was used. The MMMU benchmark is designed for large-scale multimodal models and includes a diverse dataset of 100,000+ image-text pairs across various academic and real-world domains. The dataset includes high-resolution images spanning categories such as medicine, engineering, chemistry, art, literature, and history. Each image is paired with a descriptive text that offers context, and tasks involve answering questions that require reasoning based on both visual and textual information. MMMU provides a comprehensive test of multimodal understanding by requiring models to integrate both image and text inferences for tasks such as image captioning, visual question answering (VQA), and visual commonsense reasoning (VCR).

By testing the visual cortex model using this benchmark, we aimed to assess the model's performance across a wide range of multimodal reasoning tasks, from abstract scientific concepts to artistic interpretations. This evaluation allows for a direct comparison with other state-of-the-art models that have been trained on similar multimodal datasets.

A user interface was developed using React for the frontend and FastAPI for the backend, allowing users to submit image and text-based queries to the LLM topology network. This interface enables practical experimentation and interaction with both the topology manager and the models deployed on it.

This report is structured around the components required for building a cognitive model of the visual cortex using multimodal language modelling and agent orchestration. Section 2 details the necessary background required reviewing relevant work in multimodal language modelling, cognitive system principles and agent orchestration. Section 3 introduces the preliminary design and implementation of the LLM Topology Manager and a simulation of a housefly's brain process. Section 4 focuses on the implementation of image handling modalities, finetuning of Phi-3.5-mini-instruct and implementing the visual saliency model. Section 5 reports experimental results, and reports performance evaluations on the MMMU benchmark and assessments of the saliency system. Section 6 concludes the work completed in this report.

2. Background

This section provides a brief discussion of the related work and the necessary concepts to be familiar with to completely understand the intricacies of the project. We discuss existing literature and gaps in research. We discuss works that explore the possibility of using large language models (LLMs) to implement functions of human brain regions. We also discuss multimodal language models, the rise agentic frameworks and the systems enabling agent interactions, and the relevance of simulating visual perception by modelling the human visual cortex. These topics lay the foundation for development of an agentic system that models and simulates brain processes using LLMs.

2.1 Related Work

Chen et al., [5] developed a large-scale model of the primary visual cortex (V1) by integrating various anatomical and neurophysiological datapoints into a V1 mathematical model. Their brain-like neural network model successfully replicated several characteristic visual processing capabilities of the brain. It showed robustness to noise and the ability to process widespread visual information over various sections of time. This work emphasizes the potential of data-oriented approaches in building biologically explainable models of primordial visual processing.

Wang et al., [6] explored the integration of natural language supervision into visual models to better explain the high level image processing features of the human visual cortex. The research demonstrated that models like ResNet50 trained using CLIP framework for image understanding using image-caption pairs, could explain about 79% of the voxel response variances in hold-out validation data. The model trained on this framework surpassed models trained just on image-label pairs or just plain text data. It highlights the ability of multimodal training in capturing complex neural representations.

Krassanakis et al., [7] conducted a study on visual processing of human behaviour using the UAV123 database. They recorded observations of humans watching unmanned aerial vehicle (UAV) videos. Their research aims to explain the way human attention changes when the perceived environment has dynamic changes, particularly with real-time aerial footage consumption. The authors used an eyetracking device called EyeLink 1000 Plus (eyetracking technology) to capture raw gaze data. They were able to record a variety of fixation data like saccades and heatmaps. They compiled a dataset, having 19 videos with a FullHD resolution of 1280x720px at 30fps. The high resolution and frame rate with a gaze rate of 47 seconds is useful for attention tracking during free viewing tasks (FVT). The data was analysed using the I-DT algorithm for fixation detection. It provided a valuable contribution, a real-world dataset for attention shift research real-world scenarios.

Wang et al., [8] perform a similar deep learning based video saliency prediction, an area of perception research that aims to predict humans focus coordinates and attention spans in video sequences. They developed their model by training it on a large-scale dataset (DHF1K) containing around 1000 video clips, of size 640x360px, with dynamic free viewing conditions. The dataset was captured using the Senso Motoric Instruments (SMI) RED 250 eye tracker at a frequency of 250Hz, making it one of the largest data sources for training. Their work integrating advanced models like the DeeplabV3 to improve prediction accuracy in video-based saliency tasks. This study showed the importance of spatio-temporal factors and their effect on dynamic saliency prediction. It also compiled trends with regard to the variability of visual fixation patterns across different age groups (20-28 years).

Fan et al., explored the intersection of emotional content and visual attention in their *EMOd* dataset, focusing on the relationship between sentiment in images and human visual fixation [9]. This study provided important knowledge about how emotional context influences attention allocation in visual perception. After analysing images with varying emotional tones, the researchers showed that the emotional content of an image significantly impacted where subjects focused their attention, providing a nuanced understanding of visual saliency that goes beyond traditional feature-based approaches. They too utilized a range of eyetracking technologies to capture detailed gaze patterns, offering a unique dataset for studying emotional attention in visual saliency research.

2.2 Multimodal Language Modelling

Improvements in transformer based language modelling have extended their functionality beyond text oriented tasks to include vision-language tasks such as image captioning, visual question answering (VQA), and multimodal reasoning. Modern auto regressive architectures such as GPT-4, Gemini, and LLaVA (Large Language and Vision Assistant) [14] demonstrate how integrating vision and language features into a common embedding space can enhance the model's understanding of real-world scenarios. They are built by combining pretrained language models with vision encoders, typically based on Vision Transformers (ViT) and training them on multimodal datasets like COCO, Visual Genome, or custom web-scraped data that pair images with language captions.

Developments in vision-language pretraining, seen with models like DALL-E and Flamingo, have shown how multimodal training can simulate human-like understanding. DALL-E, an image generation system by OpenAI, for example, generates new images from textual input, something that resembles how the brain can form mental pictures based on what it hears or reads. Flamingo, a multimodal few-shot learning model, on the other hand, demonstrates how fine-tuning pre-trained models with multimodal tasks can replicate the functionality of brain regions responsible for visual recognition and language processing. This shows how LLMs can be used to simulate complex cognitive processes.

Using multimodal models to replicate human brain functions has become a compelling area of research in cognitive modelling. These neural network based models learn from large datasets containing both visual and textual components. They train to perform learning based cognitive tasks like object recognition, scene understanding, and attention allocation. Recent work aimed at simulating the dorsal and ventral systems of the human visual cortex has demonstrated how using combined image – text pair embeddings in deep learning models can provide more accurate models of how the brain processes visual stimuli, resulting in significant improvements in visual saliency prediction and similar fields.

2.3 Agent Orchestration

In addition to developments in transformer based architecture, there is increasing interest and research work in multi-agent LLMs, where more than one LLMs interact with each other to solve tasks. They often draw inspiration from real life teams and how they collaborate in real life, where different parts have specialized roles but collaborate through defined and automated communication patterns to achieve a goal. Frameworks like Crew AI [15] and AutoGPT use agent-based paradigms where individual LLMs (or LLM-augmented agents) are assigned specific roles with tasks, and the agents communicate and delegate sub-tasks amongst themselves. However, they typically lack fine-grained control over their interaction topology, leading to untraceable or hardcoded communication patterns.

Frameworks like LangGraph and tools like ReAct and Chain-of-Thought (CoT) prompting are built to abstract LLMs into agentic encapsulations, have explored structured reasoning through controlled execution flows. However, these are often limited in modularity and do not support easily configurable topologies where LLMs can be arranged as nodes in a graph with defined communication rules.

The concept of agentic reasoning has evolved in parallel with the inclusion of Retrieval Augmented Generation (RAG) techniques. RAG enhances LLM performance by allowing it to pull external information from vector databases (like FAISS, LanceDB) during inference time. This enables models to remain lightweight and general purpose while still retrieving domain-specific knowledge dynamically.

2.4 Cognitive Modeling of the Visual Cortex

Cognitive modeling is the process of creating computational representations of the neuro-physiological processes that occur in the human brain. The study tries to simulate aspects like perception, memory, reasoning, and decision-making in ways that are predictive, modellable and explainable. In recent years, modeling efforts have expanded beyond parametric and simple symbolic or rule-based approaches to incorporating neural network based architectures, which better capture the distributed, iterative and dynamic nature of cognition.

An important target system for cognitive modeling is the human visual cortex. It processes and interprets visual information through a highly organized and interconnected network of regions. The visual system is philosophically divided into two major pathways: the dorsal stream, associated with motion and spatial relationships between objects in the visual field ("where" information), and the ventral stream, associated with object identification and form recognition ("what" information). Other lower-level areas such as V1 and V2 handle simpler feature detection activities like edges and textures, while higher areas like V4 and V5 specialize in color and motion processing, respectively.

Conventional vision models are built upon rule based hand-crafted features and static processing pipelines, which did not have the ability to adapt to varying requirements in terms of tasks or inputs. With the rise of large language models and agentic frameworks, it has now become feasible to simulate semantically rich, interactive, and context-sensitive requirements of the visual system. Using LLM-based agents that can specialize in different cognitive functions, it becomes possible to build modular visual systems where "brain areas" correspond to individual agents that interact in task-dependent ways. This opens up new opportunities for building both better AI systems and deeper models of human cognition.

Furthermore, in AI systems built upon neuroscience based principles, researchers have tried to mimic the brain's modular and distributed processing model by dividing the processes into modular prototypes for each system, particularly for tasks like vision. The human visual system is composed of hierarchical subsystems like V1, V2, and V4, each specialized in processing different aspects of the image captured by the human retina (e.g., edges, colors, shapes). There has been growing interest in building LLM based topologies that loosely resemble these processing units, where each LLM or model component specializes in a particular kind of reasoning or perception similar to a subsystem, and outputs are integrated using a specialized network to form a cohesive response.

Our work builds upon this line of research by combining LLM topology design, modular reasoning, and multimodal inputs to emulate how such brain-like subsystems could interact in an artificial setup. The Topology Manager introduced here provides the infrastructure needed to instantiate such networks, where each LLM is treated as a node with localized memory, definable inputs, and controlled communication pathways. This enables the construction of distributed reasoning systems that are not only modular and interpretable but also capable of handling complex, multimodal inputs — a key requirement for building systems that mimic components of the visual cortex.

3. Preliminary work

Before developing the final system, a series of preliminary explorations were carried out. This was done to better document the capabilities and limitations of current agentic frameworks in cognitive modeling. A baseline multiple drafts model was implemented using CrewAI to simulate aspects of conscious processing. Building on insights from this, a custom topology manager based on IAC principles was designed and developed to allow fine-grained control over agent interactions. To enhance its reasoning capabilities, retrieval-augmented generation (RAG) methods were integrated into the topology manager. Finally, to test the generalizability of the framework, a model simulating the visual processing system of a housefly was also implemented.

3.1 Multiple Drafts Model

We build a notebook using CrewAI to that implements Daniel Dennett's Multiple Drafts model of consciousness. This implementation will be a comparative system to the LLM topology manager which we will build to orchestrate custom agent interactions in a declarative manner. The implementation involves building three Large Language Model (LLM) agents in a team like setting, working together to analyse and interpret information. They mirror the parallel processing and integration abstractions of various streams of cognition as proposed by Dennett. CrewAI is an agentic LLM framework designed to facilitate the construction and coordination of multiple AI agents to achieve a task. Agents will be working together to solve complex tasks like LLM based data retrieval as in [4]. CrewAI's architecture is built on the concept of a "crew" - a group of AI agents that collaborate, exchange information, and realize tasks in a coordinated manner. Each agent within a CrewAI system can be equipped with specific tools, knowledge bases, and human in the loop decision making abilities. They allow for a diverse and self complementary set of skills within the crew. The agent system architecture and interaction consists of two constrained LLMs and one primary LLM:

1. **LLM 1 (Quantitative Focus):** This agent is tasked with summarizing any quantities described in the input question.

2. **LLM 2 (Logical Relationships Focus):** This agent identifies and summarizes all logical relationships or interactions present in the input question.

3. **Primary LLM (Integrator):** This agent is responsible for integrating the outputs from LLM 1 and LLM 2 to formulate an answer to the original question.

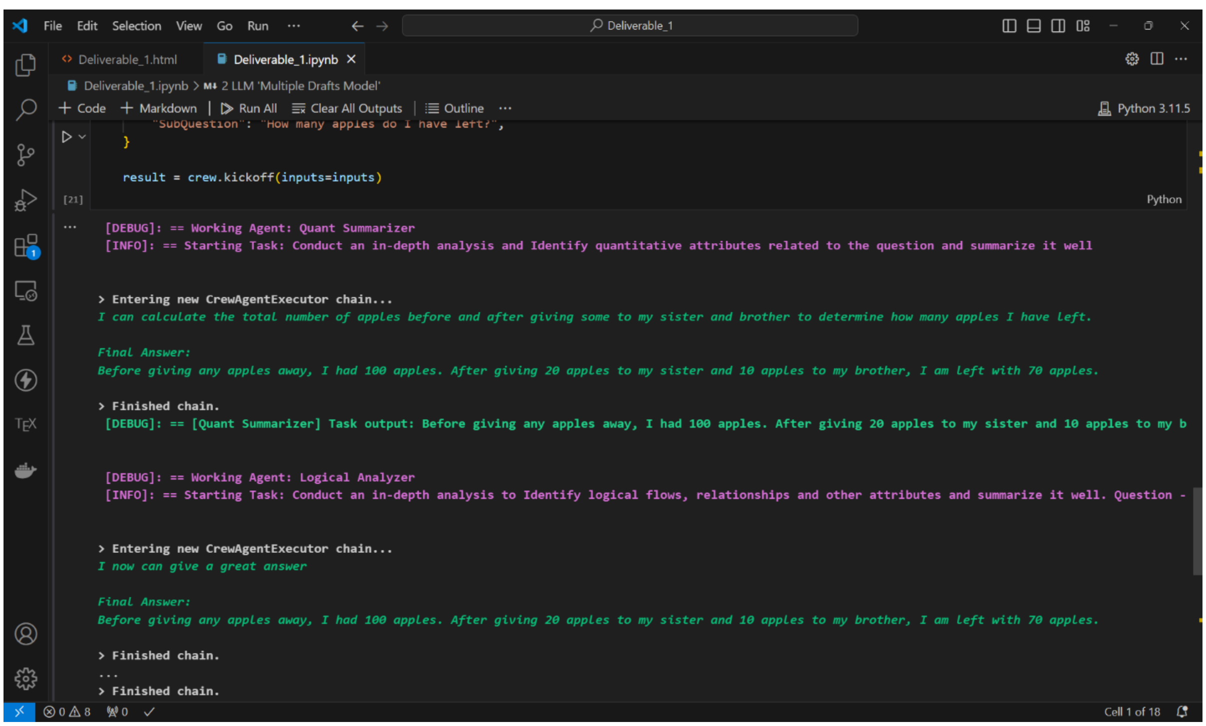

Figure 1. CrewAI Agent solving Question 1

The CrewAI framework [15] is responsible for orchestrating interactions between these agents, allowing for a centrally controlled flow of information that emulates Dennet's Multiple Drafts abstraction of parallel processing subsystems and integration of cognitive process. The outputs of each LLM for these questions provide insight into how the system breaks down and processes complex queries. This showcases the potential of our CrewAI-based implementation of the Multiple Drafts model.

Table. 1 Experimental Outputs for Dennett's Model

Question	LLM 1	LLM 2	Primary	Answer
I have 100 apples. I gave 20 apples to my sister and 10 apples to my brother. How many apples do I have left?	Before giving any apples away, I had 100 apples. After giving 20 apples to my sister and 10 apples to my brother, I am left with 70 apples.	Before giving any apples away, I had 100 apples. After giving 20 apples to my sister and 10 apples to my brother, I am left with 70 apples.	After giving 20 apples to my sister and 10 apples to my brother, I am left with 70 apples. So, I have 70 apples left.	70 apples
There are 10 crows on an electric line. If a hunter shoots the leftmost crow and it falls to the ground, how many crows will be left on the electric line?	The quantities described in the question are: 10 crows initially on the electric line. 1 crow shot by the hunter.	The logical relationships and interactions between entities in the question are: There is an initial number of crows sitting on an electric line. A hunter shoots one of the crows, specifically the leftmost one. The shot crow falls to the ground. The question asks to determine the number of crows remaining	If we only consider the quantities and interactions mentioned, it seems like 9 crows should be left on the line. However, typically, if a hunter shoots, the sound would scare the remaining crows away, which would mean no crows are left on the line. Without additional context, and considering only the quantities explicitly stated, the answer could be interpreted as 9 crows. But if we consider a realistic scenario where the	0 crows

		on the electric line after the shooting incident	noise would scare away the rest, the answer would be 0 crows left on the electric line.	
Statements: All trucks fly. Some scooters fly. Conclusions: All trucks are scooters. Some scooters do not fly. Which of the conclusions are correct according to the statements?	Analyzing the statements, we can see that all trucks are stated to fly, indicating that the entire set of trucks possess the attribute of flying. On the other hand, only some scooters are mentioned to fly, implying that there is a subset of scooters that can fly, but not necessarily all of them. Therefore, the correct conclusion according to the statements is that all trucks are not scooters, as there is no direct correlation between the flying attribute and the type of vehicle.	In conclusion, the statements do not support the conclusions that all trucks are scooters and that some scooters do not fly.	Based on the statements provided, the correct conclusion is that all trucks are not scooters, as there is no direct correlation between trucks and scooters in terms of the flying attribute. The statements do not provide enough evidence to conclude that some scooters do not fly, as it only mentions that some scooters fly without specifying the proportion that do not. Therefore, the conclusions that all trucks are scooters and that some scooters do not fly are not supported by the statements.	Conclusion 2 is correct and Conclusion 1 is incorrect

This approach not only demonstrates the practical application of Dennett's theoretical model but also highlights the power of CrewAI in creating sophisticated, multi-agent AI systems capable of tackling complex cognitive tasks through collaborative information processing but has the drawback of not being able to control the topology of the LLM interactions and suffers from a unified memory problem where all the LLMs have information about each other’s interaction which deviates from biological brain processes where specific neural processes are active for different tasks.

3.2 LLM Topology Manager

We design and develop LLM Topology Manager. It is a framework designed and built for advanced AI interactions. This framework provides us with a way to configure specifically, how the LLMs interact with each other, which LLMs they interact with and what data is shared between them. This feature is missing in current competitive systems like Crew AI. The LLM Topology Manager is a declarative framework to orchestrate Large Language Model (LLM) agent interactions by declaring them in config files in a graph based preferred topology.

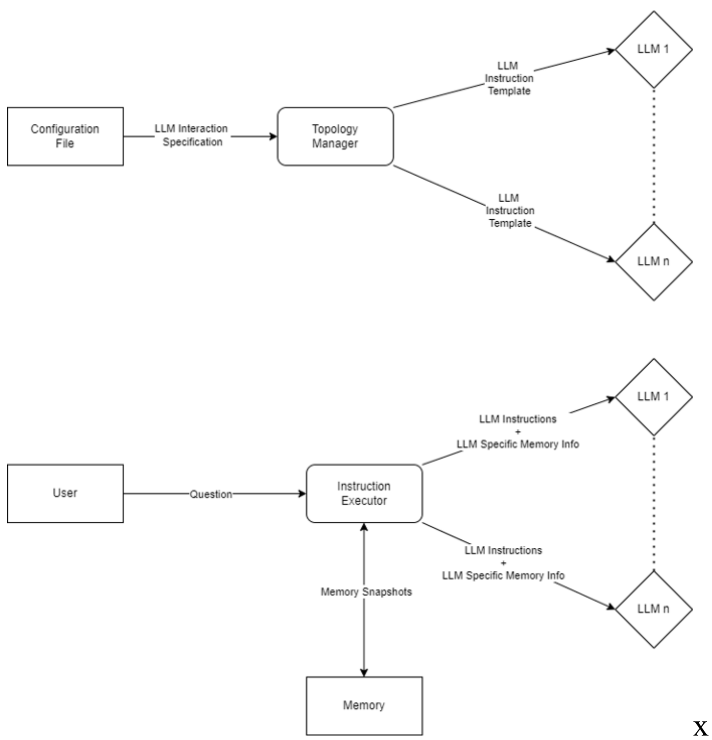

Figure 2. System Design for the Topology Manager

The Topology Manager architecture is defined to contain the following key components that interact to realize complex systems greater than any individual agent:

1. **Configuration File:** It is the blueprint file that has the agent service definitions and each of their interaction patterns in a directed graph like manner
2. **Topology Manager:** It builds the agent hierarchy and instantiates and maintains the agents. It performs logic like topological sorting to instantiate the various agents in a scalable manner and uses indegree based ordering to obtain the order of execution of different agents.
3. **Instruction Executor:** Behaves as the central processing unit, handling query processing and capturing the LLM outputs while supporting memory snapshots for context based requirements.
4. **Memory Module:** Holds historical conversational and context information for each model via memory snapshots, which is used when there is need for
5. **LLM Modules:** Abstracts LLMs into individual and platform independent agents where both on device or cloud based inference APIs can be used

The system has a module based low-level architecture, with each module having the following functionalities:

1. **Config Module**: Validates graph descriptions and configuration files to prevent missing information or cyclic dependencies
2. **Core Module**: Manages the interaction of various modules and is the glue of the framework to allow scalable and asynchronous communication
3. **DB Module**: It has RAG functionalities and has vector database abstractions to allow library independent database support for persistence and context
4. **Manager Module**: Orchestrator for agents, provides inputs and output specification for the agents and executes them in topological order while providing the necessary dependencies.

5. **Model Module**: Provides an abstraction for different LLMs ranging from cloud based inference solutions like OpenRouter, OpenAI, Gemini and local on-device variants supported on LMStudio and Ollama.

The LLM Topology Manager builds on IaC principles to enhance simple declarative configuration and reproducibility. IaC allows:

1. **Declarative Configuration:** LLM topologies and interactions are defined in human-readable files that are configurable just by modifying them in JSON format
2. **Version Control:** Configuration files provides us with versioning various config objects and helps in automated rollbacks and deployments
3. **Automated Deployment:** The system can automatically be deployed and scaled up without the need for human intervention to deploy and provide resources for LLMs and changes would be applied based on the IaC specifications.
4. **Consistency:** IaC ensures that the LLM topology is consistently and avoid human deployment errors when reproduced in different environments

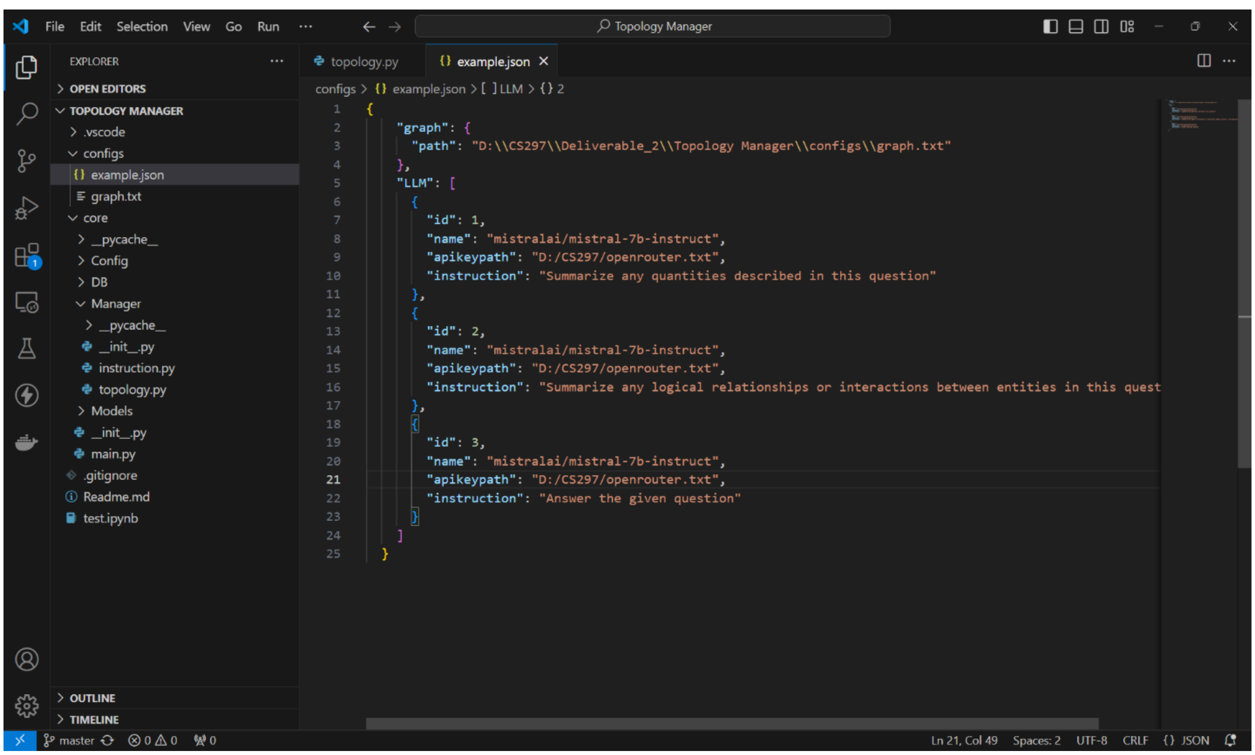

Figure 3. A JSON based IaC Configuration of LLMs

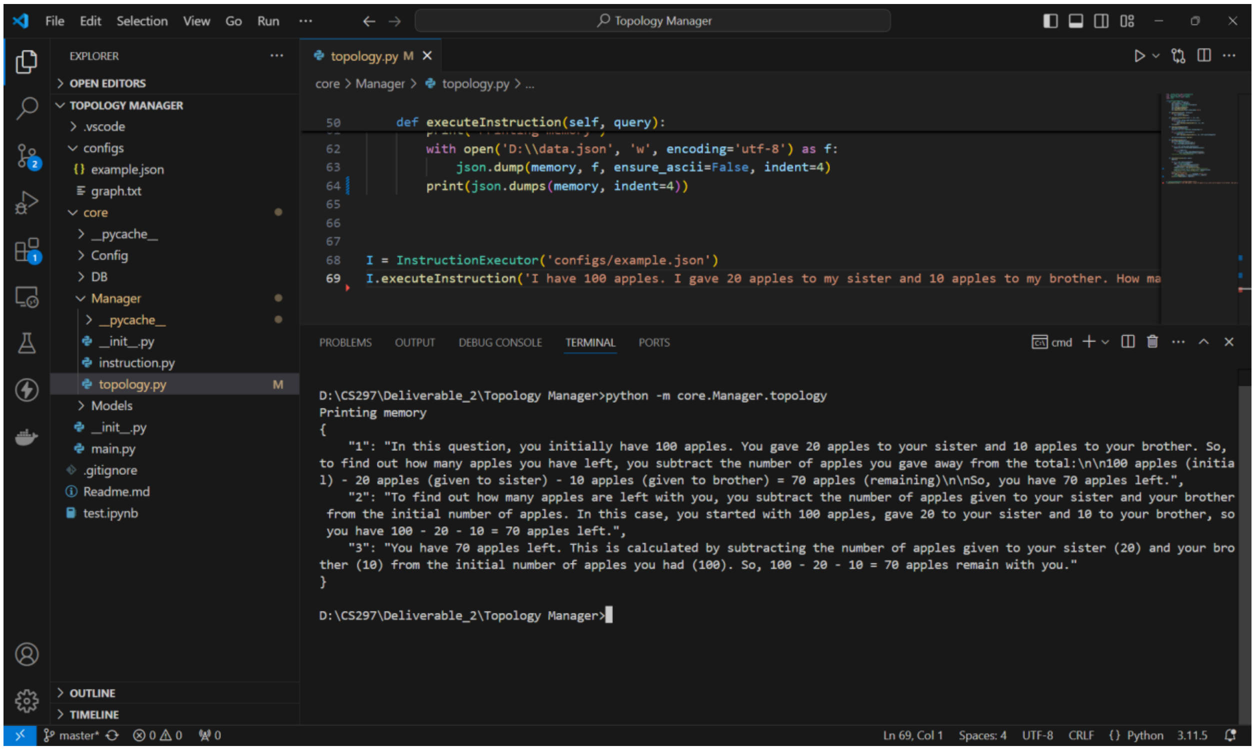

Figure 4. Topology Manager in action

The LLM Topology Manager through its LLM modularity and network flow based design offers a flexible, powerful, and efficient framework for complex LLM interactions. By integrating IaC principles and later with agentic RAG capabilities, it provides a robust solution for a wide range of AI-driven applications and research projects.

3.3 Integrating Agentic Retrieval-Augmented Generation (RAG)

We incorporate Agentic Retrieval-Augmented Generation (Agentic RAG), a technique to perform query relevant information retrieval to support inference. It significantly improves LLM performance:

1. **Query based Information Retrieval:** Agentic RAG helps extract context based information retrieval for providing the LLM with enough information to resolve the query.

2. **Contextual Understanding:** By using vector search feature of LanceDB, we can efficiently retrieve concept and query specific information that helps provide necessary context to understand the query.
3. **Improved Accuracy:** Web scraping and RAG work together to enhances the LLM's ability to provide latest and web backed accurate responses.
4. **Adaptive Learning:** Agentic RAG helps to continuously update knowledge base to consume new information and accumulating historical data.

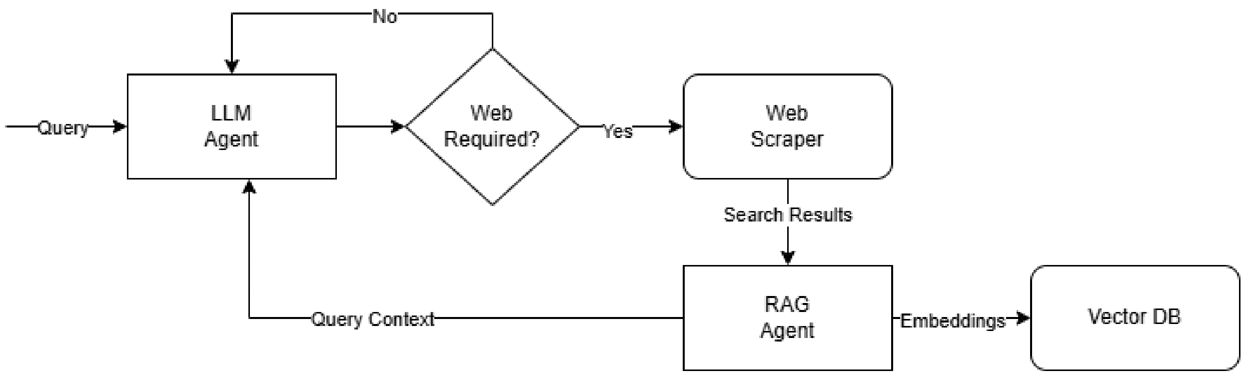

Fig. 5 Agentic RAG Pipeline

We build another agentic RAG (Retrieval-Augmented Generation) pipeline to support offline execution to allow the system to produce output without depending on cloud based LLM inference. This pipeline will incorporate the following components:

1. **Web Scraping**: We developed a python based asynchronous script based crawler to extract latest information and integrated it as a tool to conditionally crawl the web for the latest information.
2. **On-Device LLM Solution**: We used an on-device LLM using Ollama to consume the webscraped results from the vector database. Depending on the query the response quality was greatly enhanced without relying on any external cloud APIs or the need to finetune the model.

3. **Vector Database**: The web scraped data was stored in LanceDB, a multimodal vector database that runs in-memory for efficiency and allows for efficient retrieval of related information.
4. **Embedding Model**: We used nomic embed v1.5 for generating 384 dimensional sentence embeddings. It is a state of the art MLM that generates multimodal representations data, in a common text-image embedding space.

This cloud agnostic data retrieval pipeline supports the hierarchical LLM structure by providing contextually relevant and up-to-date information thus improving the accuracy of on device LLM inference. We then implement the Multiple Drafts model using this topology manager to further verify its performance and ease of defining interaction configurations with the baseline Crew AI system.

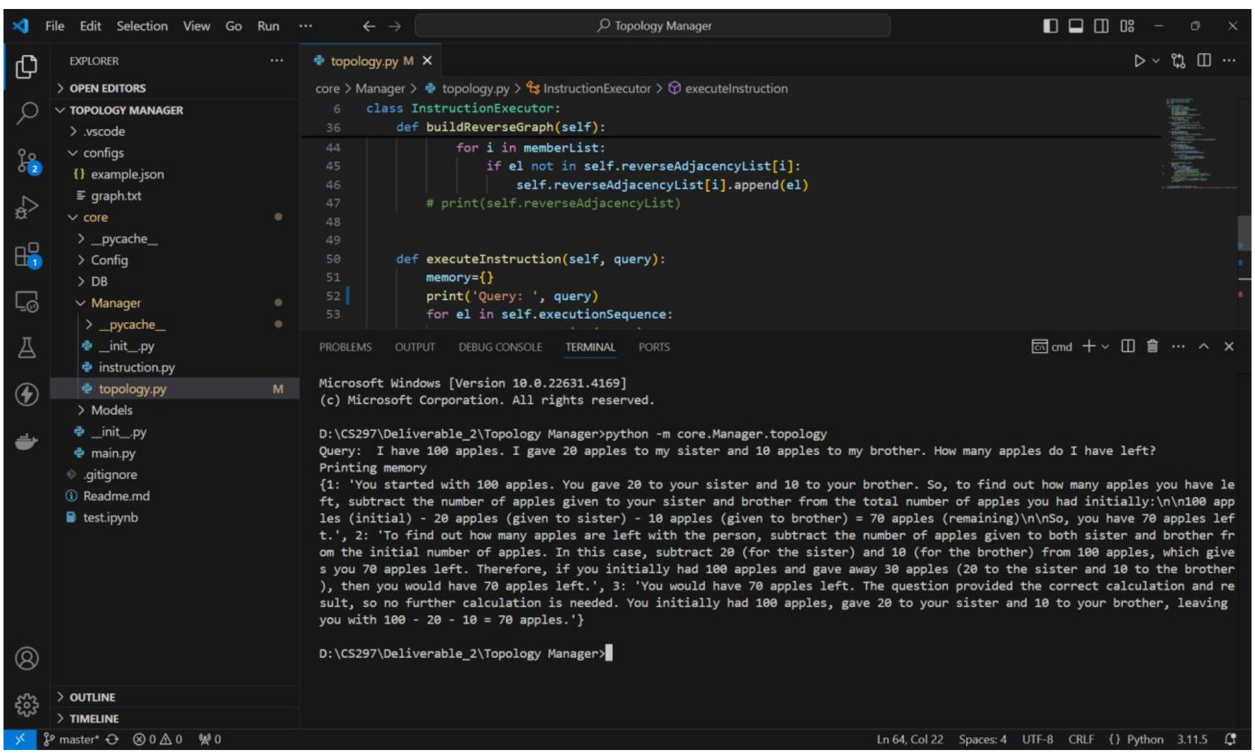

Figure 6. Outputs for the Multiple Drafts Task

The question set is the same one used for the Crew AI system. We contrast the outputs between the 2 systems. This hierarchical approach allows for a more nuanced and comprehensive analysis of the input question, potentially leading to more accurate and insightful responses. We can see that both models perform well albeit with unique but correct explanations.

Table. 2 Topology Manager Outputs for Dennett's Model

Question	**LLM 1**	**LLM 2**	**Primary**	**Answer**
I have 100 apples. I gave 20 apples to my sister and 10 apples to my brother. How many apples do I have left?	You started with 100 apples. You gave 20 to your sister and 10 to your brother. So, to find out how many apples you have left, subtract the number of apples given to your sister and brother from the total number of apples you had initially: 100 apples (initial) - 20 apples (given to sister) - 10 apples (given to brother) = 70 apples (remaining) So, you have 70 apples left.	To find out how many apples are left with the person, subtract the number of apples given to both sister and brother from the initial number of apples. In this case, subtract 20 (for the sister) and 10 (for the brother) from 100 apples, which gives you 70 apples left. Therefore, if you initially had 100 apples and gave away 30 apples (20 to the sister and 10 to the brother), then you would have 70 apples left.	You would have 70 apples left. The question provided the correct calculation and result, so no further calculation is needed. You initially had 100 apples, gave 20 to your sister and 10 to your brother, leaving you with 100 - 20 - 10 = 70 apples.	70 apples
There are 10 crows on an electric line. If a hunter shoots the leftmost	The question describes a quantity of 10 crows that were on an electric line. After the hunter	The logical relationship here is that if the leftmost crow is shot and falls to the ground, the	There will be 9 crows left on the electric line. This is because when the leftmost crow is shot and falls to the ground, it is no longer	0 crows

crow and it falls to the ground, how many crows will be left on the electric line?	shoots and kills the leftmost crow, there will be 9 crows remaining on the electric line. The information about the fallen crow or the ground is not relevant to the remaining crows on the line, hence not included in the summary.	number of crows remaining on the electric line will decrease by 1, as the fallen crow is no longer part of the group on the line. Initially, there are 10 crows on the electric line, and after the leftmost crow is shot and falls to the ground, there will be 9 crows left on the electric line.	part of the group on the line, resulting in a decrease of 1 from the initial number of 10 crows.	
Statements: All trucks fly. Some scooters fly. Conclusions: All trucks are scooters. Some scooters do not fly. Which of the conclusions are correct according to the statements?	The conclusion that "All trucks are scooters" is not correct according to the given statements. While some scooters can fly (as per the second statement), the first statement states that all trucks fly, which contradicts the general understanding that trucks do not fly like scooters. The conclusion that "Some scooters do not fly" is correct, as it is directly stated in the second statement.	Given the two statements "All trucks fly" and "Some scooters fly", one can conclude that "Some scooters do not fly" is a correct conclusion since the second statement explicitly states some scooters can fly, but doesn't say all scooters fly. However, the conclusion that "All trucks are scooters" is not a correct conclusion because, per the general understanding, trucks are not commonly associated with flying capabilities.	Based on the given information and reasoning, it can be asserted that the conclusion "Some scooters do not fly" is a correct statement as it does not contradict either of the provided statements. By contrast, the conclusion "All trucks are scooters" is not a supportable conclusion since it neither agrees with nor follows logically from both of the initial statements.	Conclusion 2 is correct and Conclusion 1 is incorrect

3.4 Housefly Vision Simulation

This task aims at implementing a sensorimotor pathway of a housefly using the Topology Manager. The goal is to implement a neurobiological process with LLMs modelling their subsystems, the inputs being stimuli (text descriptions of the environment) and the outputs fed to subsequent subsystems. This work builds a foundation and acts as a proof-of-concept for exploring the possibility of LLMs modelling more complex visual pathways of the human brain.

We implemented a housefly sensorimotor pathway using the Topology Manager framework. This implementation includes seven key subsystems: Stimulus Selection, Head Direction, Internal Drives, Circadian Rhythm Simulation, Multisensory Integration Simulation, Self-Motion Processing, and Optic Flow Simulation. Each subsystem was defined with biologically accurate context [12] , inputs and output requirement definition. These configuration descriptions collectively modelled the housefly's decision making by simplifying it into modular systems. This elaborate model provides us with a possibility of implement a similar visual processing system of the human brain that has similar but specialized processing subsystems.

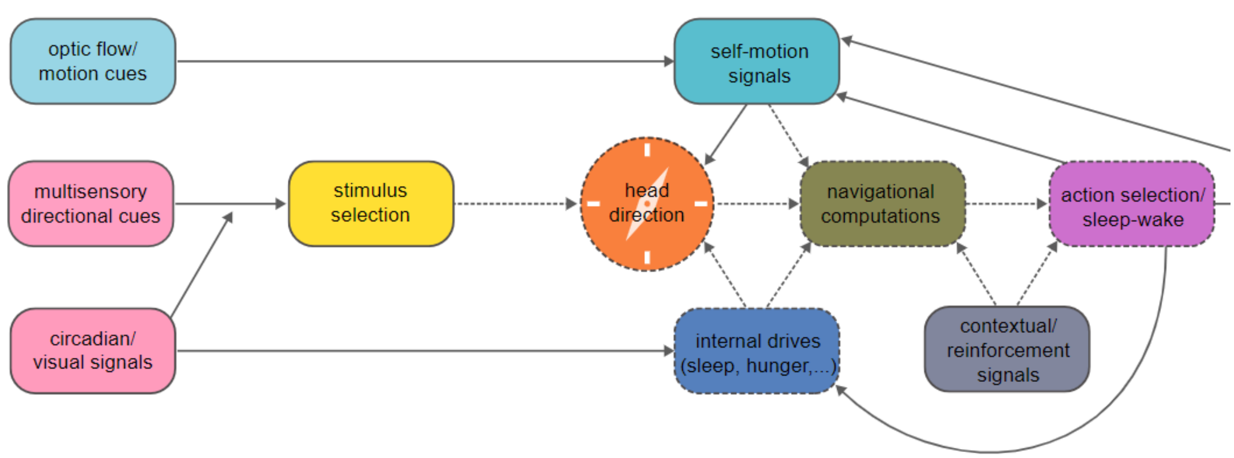

Figure 7. Housefly Visual sensorimotor pathway

To implement episodic execution of the housefly simulation, we developed a conversational memory that stored previous outputs of the agents of the model. The modification required us to store previous outputs in memory in the form of dictionaries and feeding them back to the agent. We summarized the conversation to a shorter size to avoid conversation cascading and enabling more historically aware and context based episodic generation. This feature significantly improved our model's ability to simulate the continuous, adaptive nature of insect behaviour whose actions depended on more than just the current stimulus. It provided a more realistic framework to simulate how organisms evaluate and respond to their environment and thus successfully achieves the fly's behaviour.


```
{
    "query": "Behave like you are this subsystem and generate the outputs as specified and provide one of the outputs you are created to
    "memory": {
        "4": "Circadian Signals: prepare for sleep",
        "7": "Internal Drives: seek sleep.",
        "3": "Multisensory Cues: head towards the sound, \nMultisensory Cues: orient to light source.",
        "5": "Stimulus Selection: orient to light source.",
        "1": "Optic Flow: turning right",
        "2": "Self-Motion: adjust heading slightly left",
        "6": "Output: Maintain a stable heading, adjust heading slightly left."
    }
}
```


Figure 8. Vision system output for a particular timestep

4. Implementation

The implementation involved developing and integrating several core components that collectively contribute to the cognitive modeling framework. First, we finetuned the Phi-3.5 language model to adapt it to the specific tasks required by the system, including instruction execution and multimodal reasoning. Second, we extended the instruction pipeline to support image-based inputs, allowing the system to handle visual queries through base64-encoded image integration. Third, we designed and built a visual saliency subsystem that simulates attention by highlighting focused regions in an image, helping interpret how different subsystems respond to visual stimuli. Lastly, we implemented a simplified model of the visual cortex using modular agentic systems to represent the dorsal, ventral, V2, V4, and V5 pathways, enabling us to simulate biologically inspired visual attention in a distributed, interpretable manner.

4.1 Finetuning Phi-3.5-mini-instruct

Phi-3.5-Mini-Instruct is a dense decoder-only transformer with 3.8 billion parameters trained for multilingual reasoning capabilities. The training objective of Phi-3.5 was next token prediction on high quality reasoning datasets, synthetic datasets and publicly available learning and educational resources. It is a multimodal model trained with image-caption pairs.

The implementation involved developing and integrating several core components that collectively contribute to the cognitive modeling framework. First, we finetuned the Phi-3.5 language model to adapt it to the specific tasks required by the system, including instruction execution and multimodal reasoning. Second, we extended the instruction pipeline to support image-based inputs, allowing the system to handle visual queries through base64-encoded image integration. Third, we designed and built a visual saliency subsystem that simulates attention by highlighting focused regions in an image, helping interpret how different subsystems respond to visual stimuli. Lastly, we implemented a simplified model of the visual cortex using modular agentic systems to represent the dorsal, ventral, V2, V4, and V5 pathways, enabling us to simulate biologically inspired visual attention in a distributed, interpretable manner.

We finetune a variant of the Phi-3.5 model known as the Phi-3.5-mini-instruct, which serves as a key reasoning layer within the project's overall architecture. While we demonstrated modular and collaborative LLM systems (such as the various implementation of Dennett's Multiple Drafts model), the finetuned Phi-3.5 model is intended to resolve situations where multiple subsystems produce competing outputs - drawing inspiration from the visual system in biological cognition, where varying feature detectors lead to an attention based decision mechanism.

Phi-3.5 is a family of transformer-based language models developed by Microsoft. The architecture follows a decoder-only transformer design, with 3.5 billion parameters and optimized for instruction-following, reasoning, and problem-solving tasks. The model was pre-trained on a filtered mixture of web-scale English datasets, proprietary curated data, and synthetic reasoning problems. Microsoft released the model as an open-weight checkpoint to encourage research in cost-effective and scalable language models.

Given the limited computational resources available, the finetuning process was carried out on Google Colab using a 4-bit quantized version of Phi-3.5. Quantization was necessary to reduce the memory footprint, allowing the model to fit into a single consumer-grade GPU while preserving most of its reasoning ability.

We utilized Unsloth based fine-tuning optimizations for reducing the training and memory overhead. It provides a set of zero cost optimizations which optimize training speed and reduce memory utilization without any hit to accuracy or correctness. They target VRAM usage by reordering and recomputing intermediate results without affecting the model's final output. This allowed efficient training of billion-parameter models on limited GPU support.

The model was finetuned on ReClor — a reasoning dataset composed of 6,000+ multiple-choice questions obtained from various graduate level educational exams. The datapoint consists of a prior context, a question, and a set of options of which exactly one is correct. The task is framed as requiring to select the most accurate answer. This trains the model to develop capabilities in logical inference and reasoning and it directly fits into the role of resolving from a set of decision options from vision subsystems into a single decision.

During the finetuning, the training and validation loss steadily decreased in coordination and accuracy on the ReClor validation set increased. But there were signs of catastrophic forgetting where the model began to overfit to ReClor-specific patterns. This led to a loss of generalization ability. We addressed this by aligning the model on another dataset called LogiQA, another logical reasoning benchmark built independently from a set of questions consisting of 2,442 questions similar to ReClor which were never introduced to the model.

The model's loss vs accuracy was monitored on LogiQA during the updated finetuning process. The training was stopped around 100 iterations, once the LogiQA accuracy plateaued at 72%, even though the ReClor loss showed signs of decrease. This stopping criterion was ideated to balance logical specialization and general language understanding capability, ensuring the model had its logical reasoning skills beyond only that of the ReClor dataset.

4.2 Integrating Multimodal Support

We build on the preliminary work, where we developed the Topology Manager framework. In this phase, we improved its capabilities by extending support for the construction of multimodal LLM networks capable of handling both image and text-based inputs. This enhancement fits into the larger goal of developing a visual cortex-inspired network, where different LLM agents process visual and textual information in parallel, like how the brain's visual system integrates shape, color, and context. Enabling image and language inputs is a foundational step toward this design, allowing future subsystems to reason over visual data and contribute to an integrated multimodal understanding.

Previously, we designed and developed the LLM Topology Manager — a framework built for advanced AI interactions. This framework provides the ability to configure how LLMs interact with one another, which models communicate, and what data is exchanged between them. Unlike systems such as CrewAI, which focus on agent-based workflows, the LLM Topology Manager was specifically designed to orchestrate LLMs into a user-defined topology for structured and reusable reasoning.

The LLM Topology Manager is designed to be modular. The high level system design architecture consists of the following primary components:

- Configuration File that specifies in a declarative manner, the agentic organization and collaboration rules, the backend to be used, api secrets and the interaction logic
- Topology Manager that consumes the configuration file and builds LLM instruction templates while verifying a Directed Acyclic Graph (DAG) like communication specification for guaranteed dependency availability

- An Instruction Executor that encapsulates the queries into a defined flow and upholds context through the implementation of conversational memory that contains history
- Agents themselves, these wrap various API based LLM calls, local LLM objects and RAG capable agents which process queries and generate results which is returned back for various downstream agents or as the final output.

The system leverages Infrastructure as Code (IaC) principles to have a declarative configuration setting, thus automating agent deployment, interaction and orchestration, also while integrating Agentic Retrieval-Augmented Generation (RAG) support for the agents to allow query and content based information retrieval to resolve the query.

An addition to the original design of the LLM Topology Manager was to support multimodal inputs, specifically those capable of processing both text and image-based inputs. State-of-the-art language models like Llama-3.2 demonstrate the ability to handle and understand image based inputs along with text. This requires the input images to be encoded in base 64 format before being embedded into the input for resolving queries. We write fast and asynchronous image encoders which provide encoded image results without causing bottlenecks in the inference process. The encoded image is stored in conversation history to avoid any recomputation and then passed with text content. The agent uses this multimodal input which includes text and images to generate relevant text outputs.

We implemented this into the Topology Manager's design by integrating these updates into the Agent module. We enhanced the Instruction class to handle a new input format: dict-based text image pairs with the images being optional. The query object was redesigned to contain a text field with the key name as content and a set of base64-encoded images with key imagelist. We also redesigned the Instruction Executor and Memory Module in a similar way to accommodate this format, allowing both conversation history of text and images to be stored in the same in memory cache. To handle multimodal input, we abstracted the instruction into an object structure similar to dictionaries. Initially designed to support just text, the modularity of the design let us easily extend this to the input fields to include both text and image data. The images were encoded into a base64 format and embedded into the input object during inference. It aligned with the standard technique of feeding base64-encoded images to inference for various LLM APIs. This method ensured compatibility with different vendors and cloud providers and also local models that support OpenAI type API specification to handle multimodal prompts. Thus future extensions were more modular and extensible. This also ensured that previous text based inputs could also be resolved seamlessly ensuring perfect backwards compatibility for the reasoning and conversation tasks.

4.3 Saliency Visualizer

We developed a front end using Vite and React. It was designed as a single-page application that interacts with the backend in an agnostic manner. It is built using reusable React components for the upload, image card and description components. The conversation component is responsible for uploading and sharing a user input image which would be the single input required by the backend and sending them as an API request. When an image is uploaded, it is immediately displayed in a 2 way bound preview in a hidden window which appears only when the image is uploaded. There exists a service layer within the front end that is responsible for formatting the input in the format required. It also handles multipart requests when necessary, and then parses the backend responses and displays the image.

The backend response JSON object is structured as follows

```
{
    "names": [...], // List of names of the subsystems
    "outputs": [...] // List of image-text pair outputs for each subsystem
}
```

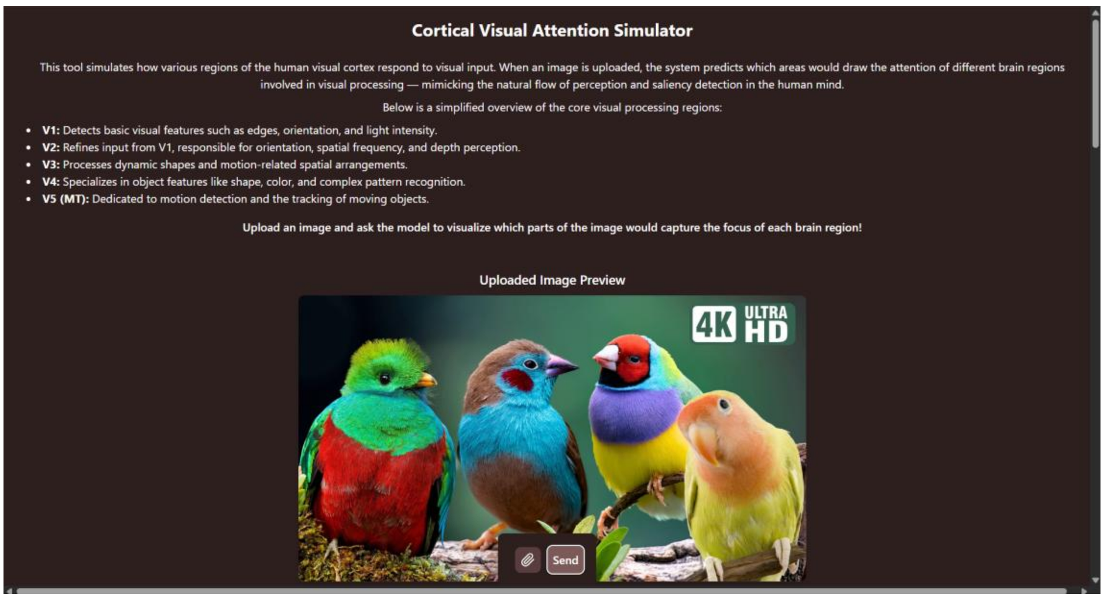

Figure 9. Screenshot of Saliency Visualizer

The output is returned as a list where each item of the outputs list includes a base64-encoded image, the name of the component subsystem that generated the image, and a textual description of why that particular region was focused on. The frontend parses this list and for each object it uses React's dynamic rendering capabilities to generate the output view. The outputs are displayed using a Flexbox based layout for responsive viewport design. Each output item is structured into a card and it includes an image, the name of the subsystem, and the explanation. A React based templating for loop maps over the outputs and builds the card for each item in the list. The images are styled to have a wide coverage of the viewport (maximum of 80% of the viewport width) with default margins and padding for readability across different screen sizes. This simple organization ensures that the interface remains uncluttered and is viewport agnostic. It is clear even when multiple outputs are shown, and it conveys to its users, a thought process and result of how different subsystems are analysing the image.

We developed the backend using FastAPI running on a lightweight Uvicorn server for asynchronous handling of HTTP requests. We also tried to support a lightweight low latency system as the core of our system was effectively slower and we didn't want to add any collateral overhead. On startup of the HTTP server, the server initializes an object of the InstructionExecutor class along with the required configuration files and graph descriptions of the communication rules. This is ready and responsible for handling and processing incoming requests. Configuration files help define the visual processing components (v1 to v6), each with its own responsibilities, the input it will get and the formatting for the output to be generated. Other peripheral health check endpoints like as /health and /uptime are implemented to monitor the service stats and verify the operation of our server.

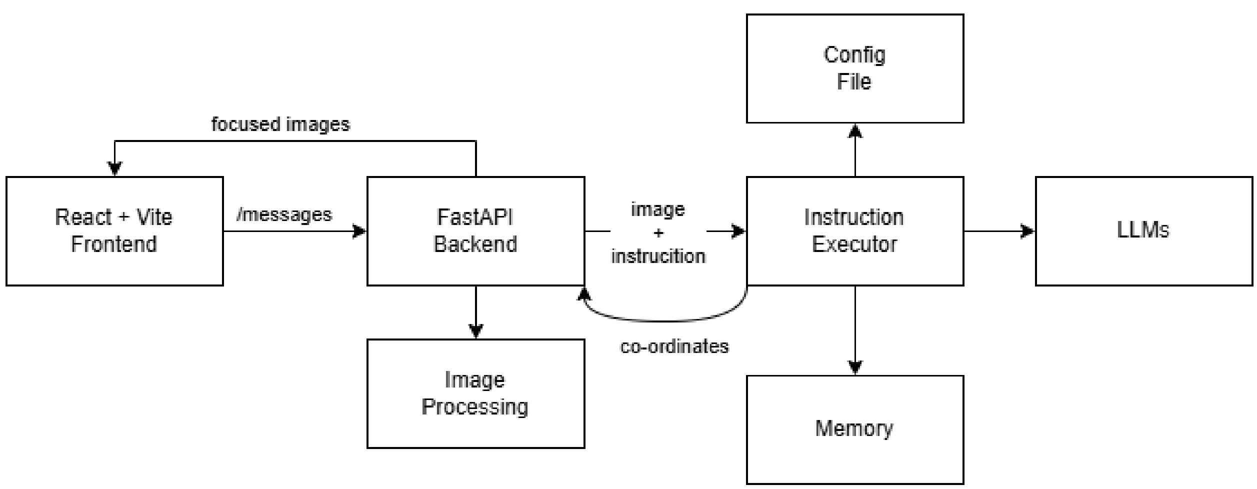

Figure 10. High Level Architecture

There is a single functional endpoint called " /messages" which is responsible for receiving the user's input image as a http request and initiating the workflow. When a POST request is made to this endpoint with an image attached, it sends it to the InstructionExecutor which is in its ready state, the instruction executor performs various validations and initializes all the agents in the pipeline and generates predictions related to what regions of the image particular subsystems are focusing on and why. Post the instruction execution step, the outputs are provided to an image processing pipeline. It designed to visualize various system's focus coordinates. For each subsystem output, a circular region with radius relative to the image dimensions is extracted based on the focus coordinates. Gaussian Blurring is applied to the remainder of the image using a mid 25x25 kernel to reduce focus in these areas and do demonstrate selective focus. The originally extracted circular region is then overlayed onto the gaussian blurred image, thus creating the effect of a region of focus to indicate the coordinates thereby simulating visual saliency.

Each subsystem produces one such processed image indicating its region of focus to visualize the corresponding subsystem output coordinates (name and textual description) and a response holding these subsystem outputs and the agent's explanation of why this region was chose. This structured output is then returned to the frontend, to be displayed in a responsive Vite and React based backend which provides a user-friendly and interactive web view of the various outputs of the different subsystems.

4.4 Implementing the Visual Cortex

We bring the components developed into a single interactive system to implement the visual cortex. We utilize the prompt driven InstructionExecutor framework that auto deploys a modular agent-task inference pattern. This allows us to declare different subsystems of the brain's visual processing network by tailoring prompts, inputs, and output formats in a controlled and scalable manner. We don't need to modify our neural architecture or finetune depending on the task. Rather we leveraged the ability of large language models by carefully crafting prompts to mirror the functions of different visual subsystems. Each subsystem was treated as a node in a directed graph based arrangement while producing its own focused output which was shared with the other subsystems to perform their functions. The outputs from each subsystem were then processed using an image pipeline to visualize attention — highlighting the specific regions the simulated brain components attended to.

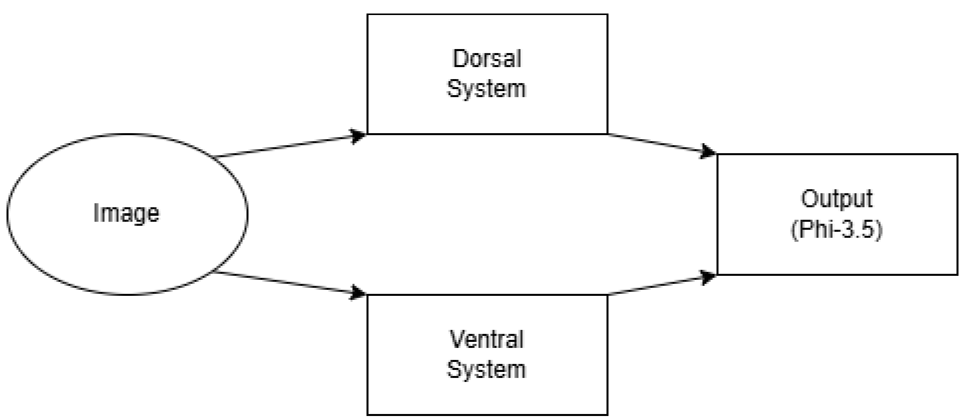

Figure 11. Dorsal-Ventral Subsystem

The first system we implemented was the dorsal-ventral model, designed to simulate the classical division in human visual processing between the "what" and "where" pathways. In our experimental setting, we defined the ventral pathway to perform object recognition and feature extraction, identifying what is present in the scene. The dorsal pathway handled spatial relationships, perspective and motion analysis to understand where objects are located and how it will change. We achieved this functional split not by changing model internals, but simply by providing distinct prompts to the InstructionExecutor. The ventral prompts asked the system to describe objects, textures, and colors, while the dorsal prompts guided it to detect positions, trajectories, and spatial configurations. This division allowed the topology manager to emulate parallel cognitive streams, staying faithful to biological systems while remaining fully modular and prompt-driven.

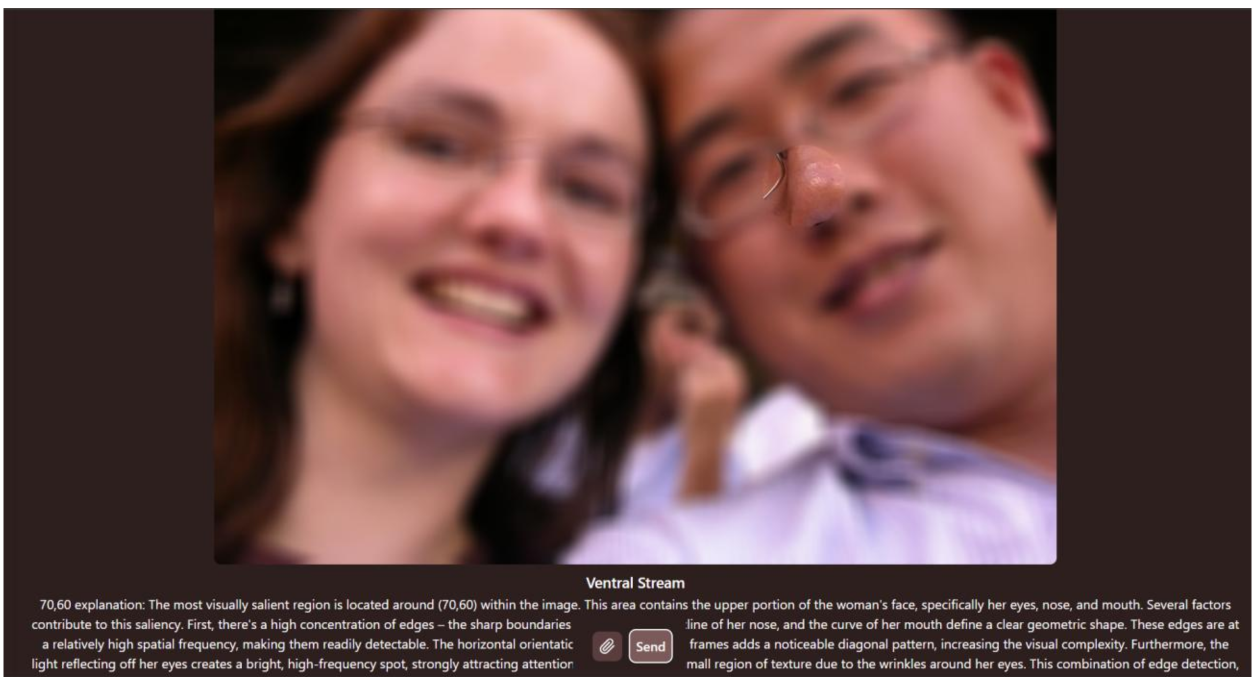

Figure 12. Output of the dorsal ventral model

Following the dorsal-ventral system, we implemented the **V2-V4-V5 subsystem** of the visual cortex using a similar prompt-driven multi-agent approach. In biological terms, V2, V4, and V5 (also called MT) form an important hierarchy in visual processing, where V2 acts as a bridge between primary visual inputs and higher-level abstractions, V4 specializes in color and form recognition, and V5 focuses on motion detection. These areas have distinct responsibilities yet are interconnected and dependent on each other, much like the division of labor between the dorsal and ventral pathways.

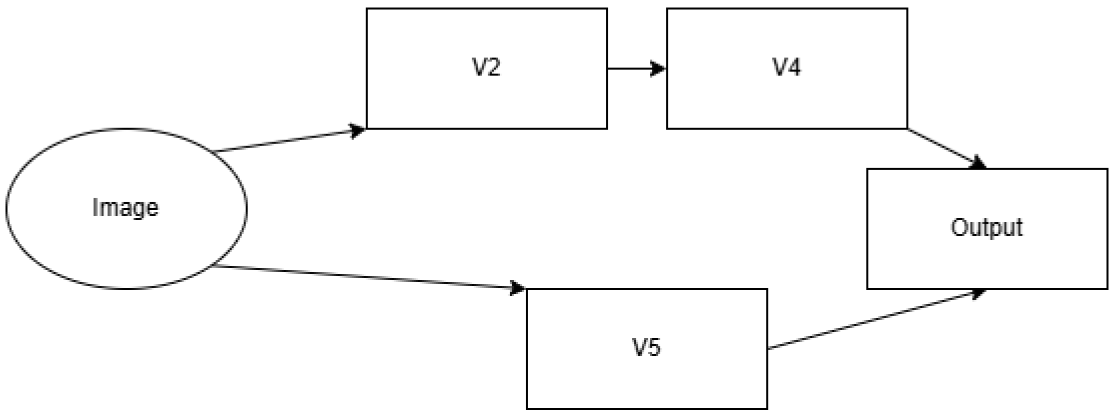

Figure 13. V2-V4-V5 system

In both simulations, the subsystems were modelled by different agents managed by the InstructionExecutor module. The V2 agent was responsible for refining and extracting visual inputs like edges, shapes, contours, anomalous regions and preparing structured information for downstream processing. The V4 agent was designed through prompts to extract rich details about colors, contrast, texture, shape and object properties to supporting fine-grained material analysis. V5 that had inputs from the V2 agent and the original inputs was focused on motion detection, object tagging, and subject tracking of various subject of the image, allowing non static aspects of the scene to be captured.

This modular and task asynchronous setup mirrors the human brain morphology where different specialized areas interact to achieve visual cognition. We ensured that the prompts had an accurate description of the tasks at hand and by keeping the number of agents and their interactions comparable to the dorsal-ventral system. This provided us with a consistent means of comparing both the simulations. The output from the V2-V4-V5 system underwent post-processing in the image domain. Each of the subsystem's generated image were modified to highlight the region it focused on, along with textual descriptions. This allowed us to visualize and verify the consistency of the coordinates and explanation of visual subsystems.

We developed the **V2-V4-V5 subsystem** in a manner similar to the dorsal-ventral subsystem and we made use of a similar architecture. In humans, V2, V4, and V5 (also known as MT) form a low-latency processing chain where V2 extracts candidate low-level visual features, V4 processes color-texture details, and V5 is responsible for motion detection and dynamic visual stimuli capture. Each of these areas have an independent role, but are also dependent on each other to create a flow of information to have a final output similar to the dorsal and ventral streams.

In our implementation, we modeled V2, V4, and V5 as individual agents managed through the InstructionExecutor framework. The V2 agent was responsible for extracting various features, patterns and shape anomalies. The V4 agent was designed to be specialized in identifying and coordinating color and object properties. The V5 agent focused on motion and optical flow interpretation. The architecture of these agents is similar to the dorsal-ventral system whereas the communication description was different: we used a comparable number of agents, a different hierarchy of processing, and the same style of prompt-based interaction to ensure a fair and meaningful comparison between the two models.

Just like in the dorsal-ventral system, each agent produced its own focused output image and a short text description about the visual region it was attending to. These results, after processing through the image post-processing pipeline, allowed us to generate clear, interpretable outputs that simulated how different parts of a visual cortex might specialize and collaborate on the same input stimulus.

5. Experimental Results

We conducted a series of experiments to evaluate the performance of different components of the system developed. This included evaluating the performance of the finetuned Phi-3.5-mini-instruct on logical reasoning and question answering tasks, evaluating the two different implementations of the visual cortex, the dorsal-ventral system and the v2-v4-v5 subsystem using the MMMU dataset to benchmark multimodal processing and understanding capabilities. We also tested the visual saliency system using eye-tracking datasets and comparing it with fixation maps. Each experiment was designed to highlight a distinct aspect of the system — multimodal language reasoning, agent-based subsystems, and visual fixation capability — thereby providing an objective understanding of the systems strengths and limitations.

5.1 Finetuning Phi-3.5-mini-instruct

We finetuned Phi-3.5 on the ReClor dataset, which is designed to benchmark logical reasoning in multiple-choice reading comprehension settings. Every instance of the data consists of a contextual paragraph, a question, and four answer options. To match Phi-3.5's input format which is in the structure of a conversation, we used a predefined prompt template by unsloth that structured each question with its context and 4 options into a dialog. This allowed the model to reason over the input in a way aligned with its original chat-oriented training distribution.

We used the supervised fine-tuning (SFT) trainer developed on top of HuggingFace's Trainer and Accelerate libraries which enabled us to perform efficient multi-GPU training and metrics tracking. The loss continued to steadily reduce over epochs, but we began noticing signs of catastrophic forgetting—where the model's general-purpose reasoning began degrading. Since our main goal was to use Phi-3.5 as an attention chaneling mechanism that interprets and integrates outputs of various subsystems (like visual modules and agent outputs), we required it to retain its information selectivity.

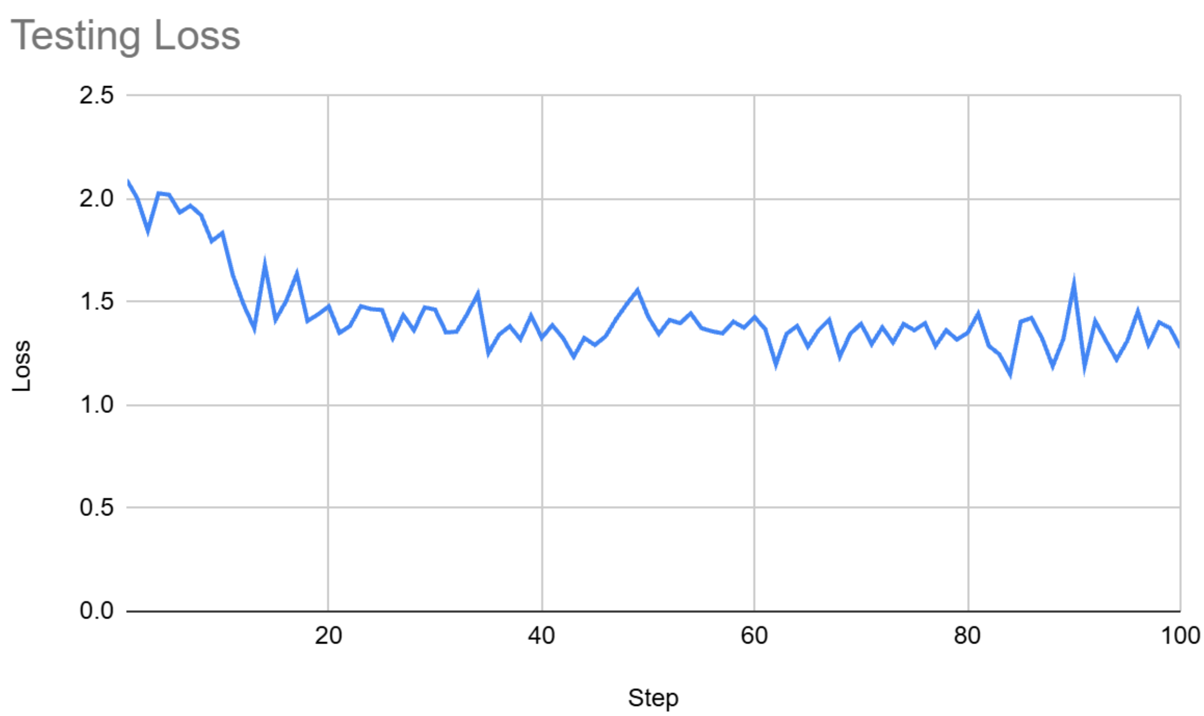

Figure 14. Testing Loss

As a result, we made the decision to stop fine-tuning once we reached 89% accuracy, balancing specialization for logical reasoning with the need to preserve its broader multimodal inference capabilities. After training, we implemented a lightweight inference module that loads the fine-tuned model and performs fast prediction over unseen examples. The model's outputs were compared against the correct answers using LLaMA 3.2 for semantic alignment and label mapping. This helped ensure robustness in evaluation even when the model's response phrasing slightly deviated from the expected format.

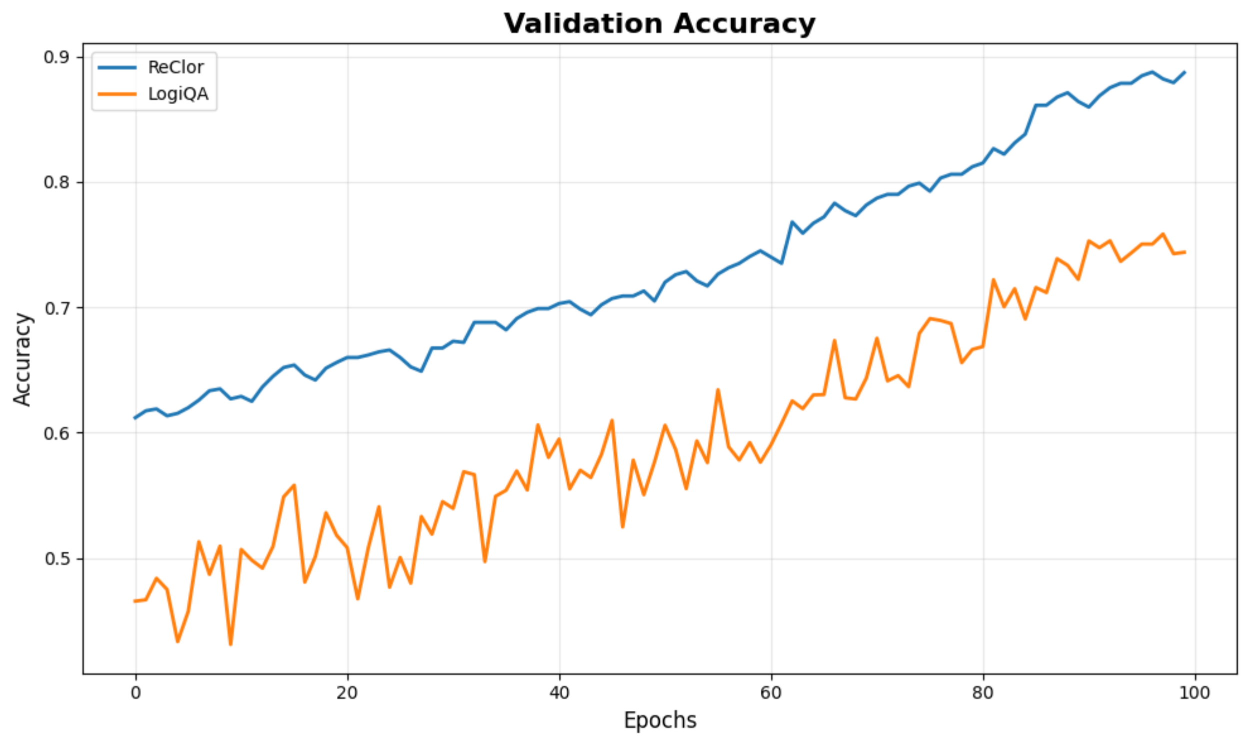

Figure 15. Validation Accuracy

5.2 Visual Cortex Experiments

We tracked two experiments to evaluate the effectiveness of biologically inspired visual subsystems in performing complex visual reasoning tasks. Both experiments were designed to assess whether modular processing, modeled after the human visual cortex, provides measurable advantages over generic multimodal LLMs.

Dual-Stream Dorsal-Ventral System on Object Distance Estimation

The first experiment evaluated a dual-stream agent that mimics the ventral (“what”) and dorsal (“where”) visual pathways. The hypothesis was that separating object recognition (ventral) and spatial estimation (dorsal) would result in object distance judgments closer to human annotations compared to a single multimodal LLM reasoning over the entire image jointly.

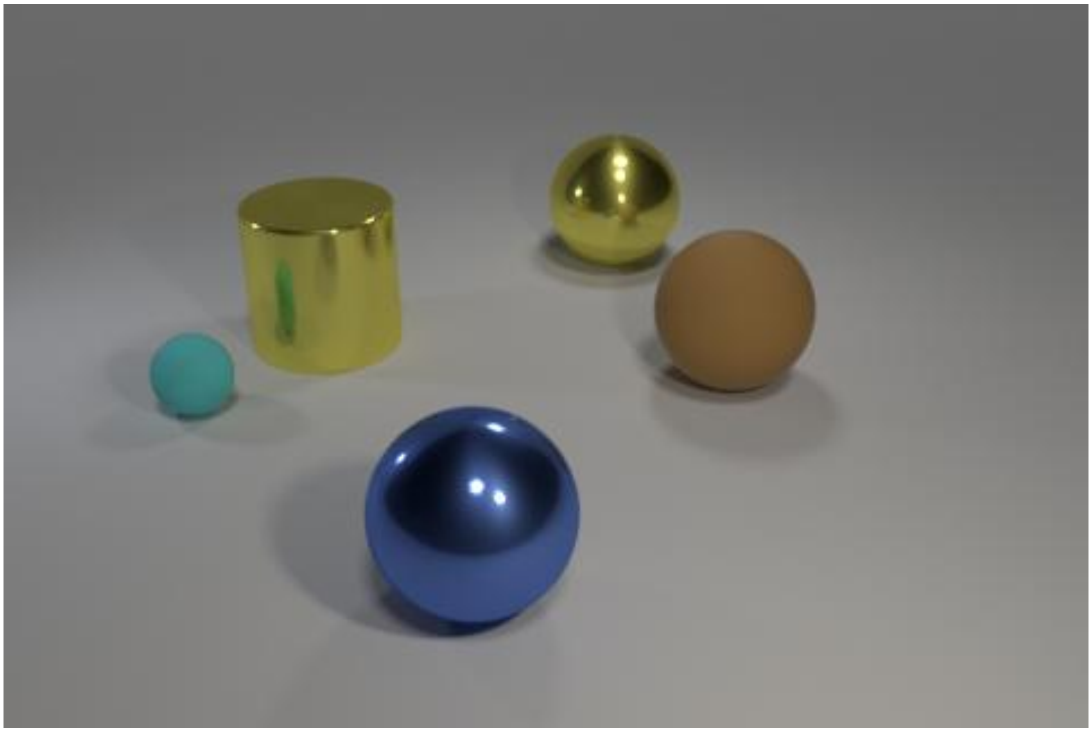

Figure 16. A CLEVR datapoint

We compiled a dataset of 100 images from the CLEVR dataset (~100,000 images) [13]. The images mainly contained synthetic 3D scenes with multiple objects annotated with shape, color, size, material, and spatial coordinates. Each model was prompted to estimate the distance of target objects. Human annotations were used as ground truth. The responses were compared and evaluated using mean absolute error (MAE).

Systems Evaluated:

1. Single multimodal LLM (LLaMA 3.2 90B)
2. Dorsal-Ventral Dual-Stream Subsystem
3. Human Annotators

The dual-stream system outperformed the single LLM, confirming that **task-specific modular pathways** improve alignment with human-like spatial reasoning.

Table 4. Agent Accuracy

Model	Accuracy
LLaMA 3.2 90B	0.48
Dorsal-Ventral Subsystem	0.67
Human	0.99

V2–V4–V5 Hierarchical System on Camouflaged Object Detection

The second experiment evaluated the V2–V4–V5 subsystem. This is capable of mimicking both low and high level visual processing in the human cortex:

- **V2:** Captures edge and orientation contrast
- **V4:** Processes shape and color contrast
- **V5 (MT):** Analyzes motion and dynamic texture

The hypothesis was that this system would detect camouflaged objects more effectively than a generic multimodal LLM baseline, due to sequential refinement of visual cues. The dataset consisted of 100 diverse images from MM-CamObj dataset [10], a multimodal camouflage detection dataset with RGB, depth, and infrared modalities, including pixel-level masks and bounding boxes for camouflaged objects. Each system received prompts containing the image and object detection tasks. Responses were recorded and compared against ground-truth object masks to compute detection accuracy.

Figure 17. A camouflage fish (MM-CamObj datapoint)

Systems Evaluated:

1. Single multimodal LLM (LLaMA 3.2 90B)
2. V2–V4–V5 Hierarchical Subsystem
3. Human Annotators

Table 5. Agent Accuracy

Model	**Accuracy**
LLaMA 3.2 90B	0.80
V2–V4–V5 Subsystem	0.89
Human	1.00

Our system outperforms the baseline model (Llama-3.2), especially in scenarios that are perception heavy. This supports the hypothesis that sequential processing systems based on the visual cortex improves camouflage object detection which would otherwise be ignored.

5.3 Testing Visual Saliency

We evaluated the visual saliency abilities of the dorsal-ventral and V2-V4-V5 subsystems by testing them on Where People Look Dataset. It is an eye-tracking dataset with 1003 datapoints where the human focus points for each image is compiled from various subjects into special images called fixation maps using an eye tracker (Tobii X50) with a 50 Hz sampling rate where the white regions indicate higher intensity of fixation. We compared the position predicted by both the systems and compared them against the ground truth fixation maps. For each input image, the backend system processed the image for both the subsystems and compared them with the saliency fixation maps to calculate the region of interest.

The predicted coordinates were used to extract the corresponding pixel values from the regions in the fixation maps and were used to calculate a weighted precision score. The idea with using precision was that the fixation maps contained more than one saliency regions and each with varying intensities. But we had a single coordinate prediction and thus could only evaluate whether our predicted value was an accurate and high-intensity fixation point and precision captures true positives and false negatives quite well. The comparison between subsystems also provided insight on how both the system's predictions varied with varying landscapes. The dorsal-ventral system which was designed to replicate the brain's spatial and motion processing pathways, was identifying general regions of interest and objects with gradients in optical flow. The V2-V4-V5 subsystem, on the other hand which focuses on higher-level visual processing such as color, texture, and detailed form perception, was better at identifying intricate features, vivid contrasts and textures and anomalies in form.

Table 6. F1 Scores of the 2 systems

Vision Model	F1 Score
Dorsal-Ventral System	0.41
V2-V4-V5 System	0.59

We computed precision scores for each system for a test split of 100 images chosen randomly from the dataset and compared the results. The differences in performance between the two systems demonstrate their respective strengths in behaving like a visual saliency prediction system. The table presents the precision scores of both systems, providing a direct comparison of their effectiveness in replicating human vision.

6. Conclusion

This research demonstrates that agent-based systems backed by Large Language Models (LLMs) can simulate cognitive processes like human vision. They offer a possible avenue for modeling the human brain's visual cortex. We built interpretable topologies based on biological subsystems such as the dorsal-ventral pathways and V1–V6 regions-the project successfully emulates core functions of human visual processing, whose tasks include contour detection, motion perception, and color analysis. The architecture of the LLM Topology Manager enables complete and declarative control over agent interactions thus overcomes the limitation in existing agentic frameworks.

The finetuning of the Phi-3.5-mini-instruct model on the ReClor logical reasoning dataset helped improve the models reasoning and summarization capabilities. A dual-stream Dorsal-Ventral agent outperformed single-model baselines for spatial estimation: it reached 67% accuracy on CLEVR object-distance judgments, compared to 0.48 with LLaMA 3.2 90B; human benchmarks reached 99% accuracy. The V2-V4-V5 hierarchical subsystem excelled at camouflage detection, with 89% accuracy on the COD10K dataset, beating the Dorsal-Ventral (0.80) and matching human-level results (1.0).The eye-tracking results demonstrate the effectiveness of the orchestration approach. The system achieving a precision score of 0.6 in visual saliency prediction demonstrates its ability to approximate human attention patterns.

Our framework facilitates systematic exploration of brain-like architectures and with suggested future work, also paves the way for future research into multimodal, distributed reasoning mechanisms that can help model multiple aspects of human cognition, like text, audio, and video processing.

References

[1] [Stanislas1998] A neuronal model of a global workspace in effortful cognitive tasks. Stanislas Dehaene; Michel Kerszberg; Jean-Pierre Changeux;. Proc. Natl. Acad. Sci. USA Vol. 95, pp. 14529-14534. November 1998.

[2] [Sattin2021] Sattin, D.; Theoretical Models of Consciousness: A Scoping Review. Magnani, F.G. Bartesaghi, L. Caputo, M. Fittipaldo, A.V. Cacciatore, M.; Picozzi, M.; Leonardi, M. Brain Sci. 2021, 11, 535.

[3] [Jeong2024] Domain-specialized LLM: Financial fine-tuning and utilization method using Mistral 7B. Journal of Intelligence and Information Systems, Jeong, C. (2024). Korea Intelligent Information System Society. 2024

[4] [Ravuru2024] Agentic Retrieval-Augmented Generation for Time Series Analysis. Chidaksh Ravuru, Sagar Srinivas Sakhinana, and Venkataramana Runkana. arXiv. 2024.

[5] Guozhang Chen et al., A data-based large-scale model for primary visual cortex enables brain-like robust and versatile visual processing. Sci. Adv.8, eabq7592(2022). DOI: 10.1126/sciadv.abq 7592

[6] Wang, AY, Kay, K, Naselaris, T, Tarr, MJ & Wehbe, L 2023, 'Better models of human high-level visual cortex emerge from natural language supervision with a large and diverse dataset', *Nature Machine Intelligence*, vol. 5, no. 12, pp. 1415-1426.

[7] Krassanakis V, Perreira Da Silva M, Ricordel V. Monitoring Human Visual Behavior during the Observation of Unmanned Aerial Vehicles (UAVs) Videos. *Drones*. 2018; 2(4):36.

[8] Wenguan Wang, Jianbing Shen, Jianwen Xie, Ming-Ming Cheng, Haibing Ling, & Ali Borji (2019). Revisiting Video Saliency Prediction in the Deep Learning Era. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1-1.

[9] Wang W, Shen J, Xie J, Cheng MM, Ling H, Borji A. Revisiting Video Saliency Prediction in the Deep Learning Era. IEEE Trans Pattern Anal Mach Intell. 2021 Jan;43(1):220-237. doi: 10.1109/TPAMI.2019.2924417. Epub 2020 Dec 4

[10] J. Ruan, W. Yuan, Z. Lin, N. Liao, Z. Li, F. Xiong, T. Liu, and Y. Fu, "MM-CamObj: A Comprehensive Multimodal Dataset for Camouflaged Object Scenarios," *arXiv preprint arXiv:2409.16084*, 2024, doi: 10.48550/arXiv.2409.16084.

[11] [Dennett1994] The Multiple Drafts Model of Consciousness. Daniel C. Dennett. In: Current Issues in the Philosophy of Mind. Oxford University Press. 1994.

[12] [Hulse2021] A connectome of the Drosophila central complex reveals network motifs suitable for flexible navigation and context-dependent action selection. Brad K. Hulse; Hannah Haberkern; Romain Franconville; J. Ruan; W. Yuan; Z. Lin; N. Liao; Z. Li; F. Xiong; T. Liu; Y. Fu. Science, 2021.

[13] [Johnson2017] CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. Justin Johnson; Bharath Hariharan; Laurens van der Maaten; Judy Hoffman; Li Fei-Fei; C. Lawrence Zitnick; Ross Girshick. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[14] [LLaVA2023] Visual Instruction Tuning. Haotian Liu; Chunyuan Li; Qingyang Wu; Yong Jae Lee. arXiv preprint arXiv:2304.08485, 2023.

[15] [CrewAI2024] CrewAI: A Framework for Coordinated Multi Agent LLM Systems. João Moura; CrewAI Contributors. Technical Documentation, 2024.