World-Leading Multi-Agent AI Systems. Made in Canada.

Instruction and overview

Innervation Overview

What Is Innervation? What Does It Offer?

Innervation is a low/no-code framework for building multi-agent artificial intelligence systems that are flexible, composable and easy-to-use. Let's break down piece by piece.

Innervation is comprised of 3 main aspects:

- Agent Customization: Users have access to many of the knobs and dials used to optimize agent calls and prompts, as well as a wide range of providers for different generative AI services and models.
- Agent Organization: Assemble agents in whatever structure you need, including branching paths, or loops. Do you want all of your agents writing to one chat, working free-form? You can do that. Do you want to construct a detailed organizational structure with feedback and hierarchies to ensure long-term plans are executed faithfully? You can do that too. You can even have agents that call other agents, or whole external Multi-Agent Systems (MAS).
- Agent Resources and Tools: Agents each can have access to not just a scratch pad, or tools, but a full virtual machine. Every digital worker can be given internet access, file i/o, and custom code environments to be optimally equipped for whatever task they might need to perform.

No-Code Framework

Users do not need any programming experience to make use of Innervation, as we have designed it to be as self-explanatory as possible. Through tooltips and detail pages, users can learn what each knob does from a practical perspective before experimenting with it. We are also introducing an agent that's been given the complete technical context of Innervation, allowing users to ask questions at *their* level of expertise. That said, power users can leverage their understanding to more quickly build up complex systems using fundamental computing concepts such as conditionals, recursion and iteration. We are open to allowing the creation and integration of custom tools and agents, but while the platform supports this, we have not yet encountered user demand for this level of complexity. Even with the level of flexibility on offer presently, power users are able to create MAS to solve problems within their organization, while non-technical users can effectively use these workflows without knowing the intricacies of their design.

Multi-Agent Systems

An agent is an entity, either living or digital, that is capable of taking input, making a decision based on that input, and then interacting with its environment based on said decision. This definition can apply to non-AI software, but typically refers to agents powered by Large Language Models or other AI systems. A multi-agent system (MAS) then is a collection of these agents working together (and/or with the user) to reach a desired outcome. Innervation makes

this communication and collaboration as effective, efficient and reliable as possible, which in turn empowers the user to focus only on crafting the system they envision.

Building For The Future

Innervation is meant to solve one problem very well; selecting, utilizing and coordinating AI systems. This involves managing information contexts, ensuring agents don't get overwhelmed (or inversely ensuring they have all necessary information), and preventing factual or logical errors from cascading across the system. By optimizing for this high level, abstract problem, we are able to solve many other, more practical problems that are of greater concern to users. Indeed, it theoretically allows for *any* (computable) problem to be solved in this way, with our ultimate hope being that we help pioneer a new paradigm of human-computer interaction. With this 'toolbox' approach we aim to unleash the emerging potential of AI, and bring into reality the prescriptive interaction paradigm we have been striving for since the original Star Trek.

Flexible

Working with Innervation takes the form of constructing each solution as a flow chart (an execution graph). Each node can be an agent or a function, and they can be linked together with fairly minimal constraints. This flexibility can be broken down into three main components; flexibility of data, of construction and of execution.

- Flexibility of data means that acceptable input and output formats cover all commonly used text (including.csv, .xls and code formats), audio, image and video formats. While some API's may need to be added to get specialized data sources, Innervation is designed with this type of extensibility built in.
- Flexibility of construction refers to the fact that there are very *very* few limits on how the flow chart itself is built, allowing for highly complex systems that can include feedback loops, batching of inputs, iterating over files/data, and conditional logic between each node. Users can even selectively show or hide the flow of information from agents to ensure they are only ever focused on the exact task requested of them.
- Flexibility of execution refers to the fact that the nodes of the constructed flow chart are capable of running any AI model, agent or function the user wishes. Indeed, they are capable of accepting, transforming and outputting any aspect of other systems built using Innervation, allowing them to be chained together, nested, or executed selectively based on external events. In practice, this means that a single MAS could be used to complete a wide range of tasks, using agents to decide which configuration of the MAS (which function, other agents, etc) should be called for a given task, and configuring the MAS as it runs to carry out that set of operations.

Together, these attributes mean that Innervation can be used to solve nearly any problem the user might face. We believe this is the next paradigm of software development and solutions architecture, and are confident that no earth-shattering advances must be made in the models themselves to achieve this vision. Indeed, all evidence we've collected in our experimentations so far hints at us not even scratching the surface of what is already possible.

Agent Customizations and Integrations

Innervation does not adhere to any single paradigm, model or provider for using generative AI. We allow for the use of any large provider of LLMs, most open source LLMs through Ollama, and preserve the deep customization options available for these models through their API, while not demanding their use to be effective. Additionally, while new tools/functions can be added, Innervation ships with a comprehensive suite of integrations, from the mundane task of reading a file's contents into the agent regardless of its format, to comprehensive statistical analysis and data science tools, and of course tools for Retrieval-Augmented Generation (RAG). These tools have a very replicable format, and we hope to make creation and testing of new tools one of our first self-automated processes.

Composable

Composition in this context refers to the ability for different components or elements to be combined or connected in various ways to create larger, more complex systems or structures. This ability to easily abstract away details allows for even highly complex systems to be used effectively by all users, regardless of technical ability. While it is self-evident that this is core to Innervation given how MAS are created, we have pushed this concept even further. Since MAS are ultimately still functions, and can also execute functions, it is possible to recursively build larger and larger systems calling MAS as functions, and then integrating these functions into 'higher order' MAS. While this does lead to significant slowdowns, Innervation is not the source (it can handle this recursion depth), rather it is the army of agents that are generated, called and must be waited on. We have not yet found when this approach stops leading to better results for open-ended tasks such as academic research, and believe we can push this idea even further as models become more efficient.

Why a Multi Agent System Builder Approach?

Accuracy and Reliability

LLMs and their adoption is hampered by hallucinations, the generation of plausible but incorrect outputs. A multi-agent system approach like Innervations can demonstrably improve the reliability and transparency of outputs by a significant margin simply by having agents check each other's work. On top of factuality, breaking a task up into parts makes each more likely to reliably execute their task, even when given detailed instructions. For example, one agent could collect information, one could verify it, and the last could format the output to user specifications.This also makes error tracing and correction extremely transparent, as each step in the system can be traced independently to identify errors, and whether they were resolved or if they snowballed, it will be traceable in natural language..

Complexity

While models are becoming more and more capable month-over-month, there are still significant deficits, which Innervation is able to mitigate with its approach. This doesn't just cover some of the weaknesses of LLMs however, it also amplifies their strengths as well. We are able to not just tackle simpler problems more reliably, but far more complex tasks than would

otherwise be possible. As models improve, error correction and explicit instructions to elicit the desired reasoning will be needed less. This will cause the scope of problems that are feasible for Innervation to solve to expand exponentially. Additionally, many models are emerging that are specialized for specific tasks or sets of tasks. We will be able to slot them into the appropriate job, while insulating them from off-policy work that they would be less exceptional at.

Extensibility

Innervation is designed for easy and rapid construction of complex workflows, but is itself designed to be iterated and improved on over time. While this was spawned from basic software principles, it has evolved with Innervation itself, with each new iteration not only being better, but itself more improvable. This is critical in a domain as fast-paced as AI, since adding new models, agent types, prompting approaches, and even entire new frameworks such as for scaling, or benchmarking, must be as easy and natural as possible to keep pace. We aim to be the antithesis of being ‘locked in’ to a single solution or way of constructing a solution, allowing for everything from tweaks up to complete reworks of any given solution with minimal additional overhead. Innervation’s universal applicability and extensibility mirrors that of AI itself, and this symmetry is the key to fully leverage AI advancements, rather than stuffing them in a box of constraints due to lack of imagination.

Adaptability

Gone are the days when in order to even start seeing value from newly implemented systems, old systems and data would have to be overhauled, cataloged and prepared. While the easiest results may come from structured data, where MAS can be built to reliably operate on it using traditional forms of retrieval, Innervation can deliver value even when primarily working with unstructured, disorganized and even messy data. This is the natural user experience improvements that generative AI allows for, acting as a ‘softer’ interface between the user and the data. It can also be used as a powerful tool in its own right for creating and maintaining internal digital libraries, converting unstructured and messy data into more human-readable forms. In this way, users can begin benefitting from Innervation immediately, and will only see that benefit increase as it helps bring order to the information ecosystem it inhabits. With this ease of integration, and the capability to remain on the cutting edge of AI developments, Innervation is the last AI system you will ever need.

Technical Capabilities

Interfaces

Innervation has four primary modes of interfacing with the user. The first two are found within the Innervation platform, the third is accessed by calling the API endpoint within your key, and the last is wherever you need/want it to be.

MAS Builder

The MAS builder uses four main types of nodes; input, output, function and agent, plus a merge node, and an agent sub-type called “XORSplit” which controls looped behaviour. Input and Output nodes allow the user to add or retrieve text or files of all major formats covering text (including tabular and code), image, audio or video. There can be multiple input nodes for multiple or segregated information streams, but there is only ever one output node (though many things can be output to a single node).

Function nodes accomplish specific transformations using traditional programming functions, with a dynamic UI to present the user with fields to fill in the required parameters. Some built-in functions include converting outputs to specific file formats, data cleaning, image manipulation, and even calling other (more narrow) AI models. Both functions and the tools used by agents are defined using the same code (build once, run anywhere), so any function can be either standalone, or used by an agent. The manner in which they are used is largely up to the specific situation; consistent and reliable execution of the same transformation is easier to do with functions, while situations with less predictable inputs are better served by agents with tools who can choose at runtime what to use.

Agents are the last and arguably most important type of node, and this is where the iterative pattern (how the agent loops over subsequent calls), the underlying model used, tools, parameters, and prompts are configured. Agents receive input from all edges, but output to each edge individually, allowing for multiple messages or transformations depending on who/what the information is being sent to. The exception for this is the XORSplit node, which is very similar to an agent, but will only output to one edge and represents a very advanced form of flow control.

Chat Interface

This interface allows the user to talk to different agents/MAS with different chat histories and models as they would ChatGPT or Claude. Some agent types are specifically designed for this use case, allowing them to give the impression of a single agent, while also coordinating the rest of the MAS in the background to accomplish user defined goals. Innervation’s chat interface boasts the same set of tunable parameters and capabilities, with the addition of variable chat histories and MAS that can be utilized interchangeably. This gives users who aren’t concerned with the design or construction of MAS a familiar and simpler way to interact with them that still retains all their higher capabilities such as tool use, RAG and their multi-agent nature. This is particularly relevant for tasks that innately require a human participant, such as tutoring or a personal assistant MAS.

Innervation API

While the MAS Builder is useful for constructing and testing MAS, actually using them day to day is not made easier by requiring users to interact with a single, static interface. For this reason, we have an API that makes it extremely easy to run MAS programmatically, allowing inputs to be defined, and outputs to be captured and shuttled as desired. Indeed, even some components of Innervation that don’t need access to the details of execution and thread

management are implemented using the API. For example we call the API from our benchmarking code to keep things clean, transparent, and repeatable, all of which is critical if our benchmark implementation is to be trusted.

Cellular MAS

The final interface is an extension of the API, but one we hope to make easy enough to implement that it can be counted as a separate system. We are tentatively calling them MAS Cells or MASCs (pronounced like ‘mask), and they are simple Dash interfaces that can be created by a dedicated MAS to fit both the specifications of the underlying MAS being used, as well as the user’s requirements. They can then be used as standalone applications that run in the background and execute the MAS when triggered by events on your machine, server, site or device. While these interfaces CAN swap between different MAS, we believe they will primarily remain static and are an easy and portable way to use workflows. We will begin by productizing a few MAS using this method, but hope to make it available to users if the framework proves successful and scalable. Ultimately, we hope that both MAS and Cell UI’s will be the objects of trade in a multi-agent market place, which we see as delivering more substantive value then the GPT store and other nascent AI marketplaces.

Agent Customization Options

Agents can be customized in the following ways, from any interface:

Model

Models like those from Google Deepmind or Open AI can be selected, and include all major providers as well as open source and custom models. Users can even add new versions or instances of a model that has been fine-tuned on their data or for their purposes.

Agent Type

Different methods of encouraging the LLM to reason are constantly developed, and some such as “Chain of Thought” reasoning have led to significant improvements in agent capabilities and reliability. Innervation easily and quickly integrates such developments so that users can benefit from new approaches soon after they arrive.

Input Text

The text the agent receives, either from another agent, or the user. This is the prompt the model will work with during a single call, and it will change at each step as the conversation or MAS progresses.

File List

Analogous to the Input Text, but is a list of files the model will ingest, given by the user or other agents. This can contain text files, but is more for sending images, audio or video to the model. Either the File List or Input Text can be empty, but at least one must contain data for the agent to use.

Max Iterations

Agents, at their most basic, are requests to a model in a loop such that their output is fed back into their input to continue their ‘train of thought’. The max iterations parameter dictates how many of these loops a selected agent can undertake, allowing you to constrain agents that may take too long to get to an answer, while giving others the ability to iterate as long as there is some improvement in the output each time.

Final Iteration Prompt

This prompt is inserted when the ‘max iterations’ is reached or the agent has decided to give their final answer, and can be customized for each outgoing edge, allowing for different configurations and formats to be given.

Model Parameters

Four different model parameters can be adjusted to alter the model output and context:

- Temperature: The variability or ‘creativity’ of the model
- Top-P: Controls the vocabulary of tokens that can be generated, similarly results in more or less ‘creative’ outputs
- Seed: The Random seed used to initialize a model’s run, will lead to more consistent outputs and very useful for testing
- Output Pages: The number of pages to output (roughly 250 words per page). This is leveraging a proprietary sub-system we’ve implemented that allows models to give responses of a consistent (and often extended) length compared to raw prompting approaches.

These parameters allow for more or less deterministic output, as well as giving fine-grained control over how much context it needs, and how verbose it should be in its response.

Chat History

The history of responses given to the agent as context. Switching the chat history in essence switches out the ‘memory’ of the agent, and therefore the context of what it knows about what it is doing. This is an easy, intuitive way to deliver the context a given agent needs to complete its task.

System Prompt

The message prepended to every prompt the user or other agents submitted as input. This can be thought of as the persistent ‘instructions’ dictating what the agent will do with its input. For example, the system prompt could get an agent to give its response in French, or iambic pentameter, or stating that it is a highly competent web developer (thus improving its dev-related responses).

Agent Prompt

The prompt given to define the agent’s overarching behaviour. This does not need to be altered in most cases, as it is already optimized for each agent type, but is available if changes to that behaviour need to be made.

Tools

Tools such as web search, code execution, or an API integration the agent can call during its run. These tools are functions written specifically to be used by agents, and allows them to interact with the digital world in far more interesting and impactful ways.

Collections

Vectorstore retrieval, usually called Retrieval Augmented Generation (RAG), uses ‘embeddings’ (lists of numbers) to query collections based on how similar the embedding of the query itself is. This form of search relies on the context and meaning of a document, rather than matching exact words, allowing for more precise and accurate retrieval of information. This allows the agents to understand user documents, files or databases, or even search entire websites much more meaningfully in the context of answering queries, allowing agents to respond more reliably.

Advanced Options

At present, we have 4 ‘advanced’ features which further enhance the configurability and capabilities of MAS when used appropriately.

- **Enable Prompt Optimization**: uses the ‘Respond and Rephrase’ or RAR technique to enhance the user’s input prompt. This can also be done collaboratively with the user.
- **Pre-Run RAG**: does a RAG query of the selected collections *before* calling the agent, presenting the results as initial context. This is useful when you already know what information it should have, or when it should always have the same info from the collections. This does not preclude the agent using RAG tools during execution.
- **Launch Control**: Toggling this on will lead to faster and more responsive performance at the possible expense of quality. Typically used only for MAS that must be extremely quick.
- **Enable Stateful Agents**: Stateful agents will retain their context and thought history, even after the execution has moved on to other agents. This is principally used for ‘manager’ style agents that must have complete historical context of their execution in order to move towards an end-state over multiple calls back and forth with other agents.

MAS Construction and Execution

The MAS executes starting from the input nodes, of which there must be at least one, and terminates at the output node, of which there must be exactly one, though it can process any number of outputs the MAS may have. This list of outputs can be accessed individually as needed to ensure they end up in their proper place. MAS can also contain cycles, and indeed this is a critical feature in allowing for self-correction, but there must be a valid path from every input node, to the output (no orphan nodes or dead-end paths).

Our implementation of the scheduling algorithm allows all ‘ready’ nodes to be executed in parallel on our servers, with a node being ready whenever it has all necessary inputs. This cascade continues, with each ‘time step’ executing all nodes that have received their required inputs. We are also able to ensure that so long as the user adheres to the few stated constraints that exist, they will end up with a MAS that will terminate properly (no dead-ends or infinite loops assuming the model calls succeed).

Some additional features of MAS execution that Innervation provides include the ability to allow functions to output in batches rather than all at once, a multi-channel data management schema, and advanced tools that allow agents to dynamically alter MAS (with permission of course). Even excluding the use of functions, MAS are Turing complete, meaning that they can theoretically solve any computable problem. This is not a watered-down toolkit with limited applicability; it is a comprehensive solution that will serve non-technical users as well as empower those with technical skills.

Innervation Variable Scripting Language

As mentioned previously, we're in the business of optimizing communication between agents within multi-agent systems, and thus much of our focus is on changes that 'take away' from the experience: frustration, confusion, etc. By making the system itself as robust, flexible and traceable as we can, we make multi-agent systems easier to work with, integrate and update.

However, this is one tool we've developed that, while it does increase complexity, we had to share with users as it breaks MAS free of their final shackle: their static graph nature. Innervation Variable Scripting Language (IVSL) is a simple grammar that allows users to pass not only text and files, but variables and functions as well. For example, instead of setting an input to read the entirety of its input, which may not always be 100% relevant, you could pass:

{{re.search(pattern, source, group=0)@}}

Here, {{<content> @}} indicates the use of IVSL. Source is the source symbol being evaluated, and we are searching for a specific sub-pattern. IVSL can also be used to dynamically change the graph. So, for example, you could pass the example below as input to a function node.

{{MyFunctionName, param1, param2 @}}

Regardless of the configuration of the function before, this will cause it to execute the function it was given (using the parameters). Using this grammar, we can hide, show, and evaluate information, or even change the execution of the graph, all during runtime. While IVSL is optional, and can be safely ignored while still benefiting from using Innervation, it gives users an incredible amount of control. While the edges of the graph remain important for ensuring proper execution, IVSL frees data from the same constraint, allowing for adaptive graphs that change themselves (in a limited capacity) in response to input it receives.

Unique Selling Points

Innervation provides value in two distinct ways. The first is by allowing for novel value creation for clients with deep research needs. The second is by directly targeting underserved markets and use cases using bespoke MASCs for each.

Deep Research Value Creation

Innervation stands apart from the competition in its ability to drive value for companies with deep research needs. The no-code framework empowers non-technical users to leverage the latest models, agent types, and research advancements by composing them into powerful multi-agent systems via Innervation's intuitive interfaces. But it goes beyond simply providing access to cutting-edge AI capabilities.

A key differentiator is Innervation's support for cycles and recursion within multi-agent systems. This closes the loop on tasks that have definitive and validatable answers, allowing MAS to iterate, test, assess and improve in a largely autonomous manner. Research, which is typically more concerned with provable outcomes, will be able to take full advantage of this to massively increase their efficiency and ultimate capabilities. Indeed, other systems such as OpenAI's O1 have a similar underlying philosophy in their training and architecture, and we are beginning to see the performance divide open up between subjective and objective tasks.

But continual improvement isn't just something we can set our agents up to do, it is a philosophy at the core of the platform itself. Our team is dedicated to constantly integrating and iterating on the underlying technology, ensuring that clients always have access to the most advanced agentic coordination. Crucially, this is achieved without breaking backwards compatibility, so existing solutions remain functional even as the platform evolves. In addition to leveraging external advancements, Innervation can also help conduct its own cutting-edge research on multi-agent systems. With the ability to swap both MAS and benchmarks at will, we are able to concretely show that we are delivering cutting edge performance, and using that to derive novel benefits that emerge from our experimentation. By pushing the boundaries of what's possible with MAS architectures, Innervation enables us, and our clients to stay three steps ahead of the curve, and tackle challenges others fundamentally cannot address.

In summary, Innervation's unique value for deep research stems from the combination of its accessible low-code framework, support for advanced MAS design patterns, integration of cutting-edge techniques, and a robust yet adaptable architecture. This empowers organizations to push the boundaries of what's possible with AI, into self-improving virtuous cycles. Pipelines built with innervation will not be discarded and deprecated over time, but will instead improve and adapt as needs change.

Direct Targeting of Underserved Use Cases

Innervation's architecture also makes it viable as a direct-to-consumer solution for common and well-worn workflows, channeling the flexibility of the platform into performance that rivals or even exceeds products purpose built for the task. We will be able to directly address underserved use cases and communities by working with stakeholders and developing MAS to precisely address the relevant problems they face, and rapidly iterate on solutions to ensure a perfect fit.

For example, individual MAS could provide:

- A highly sophisticated Excel assistant for every employee
- Autonomous code testing within software repositories
- Personalized data aggregation to overcome departmental information silos
- Large scale corporate data analysis and modeling
- etc.

Since *building with AI* is the problem we solve, Innervation makes it economical to develop and deploy these highly tailored solutions at scale, with the core platform streamlining the process of composing the requisite functions and capabilities. This positions Innervation to capture significant value across a long tail of industry-specific applications that are neglected or infeasible with current approaches.

In summary, Innervation's deep research value and ability to serve niche use cases both stem from its unique low-code, modular, and extensible framework for composing flexible multi-agent systems. This architecture enables unrivaled access to state-of-the-art AI capabilities, empowering a wider range of users to solve complex domain challenges without getting bogged down in code.