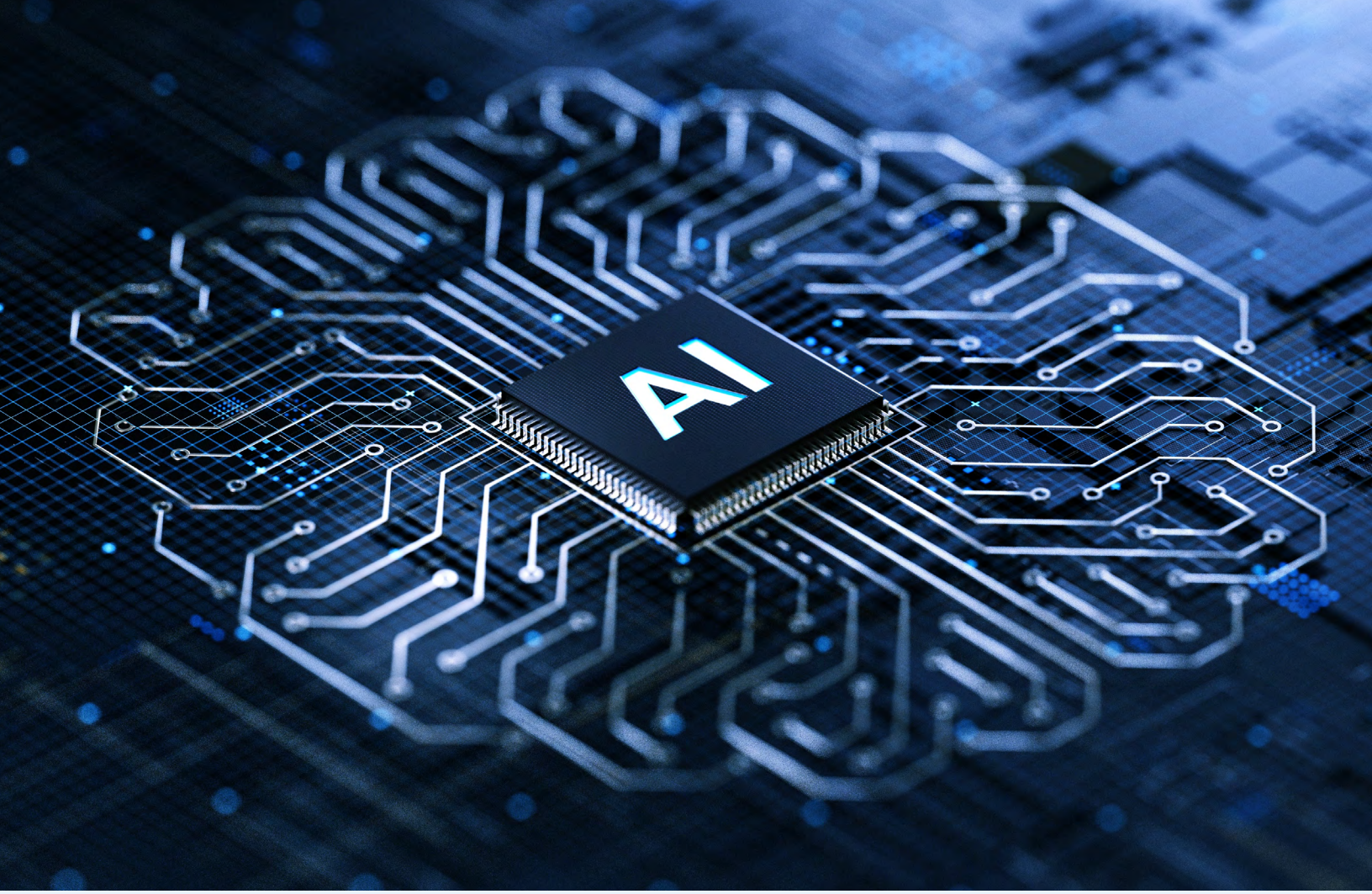


Artificial Intelligence (AI) 101: What You Need to Know

by Artificial Intelligence in Engineering Working Group

April 2026

ospe.on.ca

ONTARIO SOCIETY OF PROFESSIONAL ENGINEERS

Acknowledgements

This report was developed through the collaborative efforts of dedicated members of Ontario's engineering community. OSPE gratefully acknowledges the contributions of the following individuals:

Authored by:

Paul Longo, P.Eng, M.Eng, PMP, MMAI

Colleen Shannon, P.Eng., LLB, LLM

Dr. Jacqueline Csonka-Peeren, P.Eng., MASc, MBA

Emanuel Corthay, P.Eng.

Raimund Laqua, P.Eng.

Contributors

Aaron Pereira, MASc

Arjan Arenja, P.Eng., MBA, ICD.D

Claude Nnadi, EIT

Maria Becerra

Ryan Maclaughlan, P.Eng.

Table of Contents

Overview

The **Ontario Society of Professional Engineers (OSPE) Artificial Intelligence (AI) in Engineering Working Group** was established to educate and empower engineering professionals to use AI effectively, ethically, and responsibly in their daily practice.

As a first step, the working group developed this **AI 101 Guide** to provide a starting point for any practicing individual to learn more about AI and feel empowered to ask appropriate questions about how AI should be applied. This guide draws on expertise from members of the working group as well as open-source resources.

At the time of this guide's creation, engineers have been called on by Canada's federal government to build major infrastructure at an unprecedented speed. AI is a tool engineers can use to meet this moment.

Additionally, the federal government has increasingly invested in sovereign AI. To effectively leverage Canadian-made and owned AI, Ontario engineers should build their knowledge and skills in applying AI to incorporate it into their practice effectively.

The aim here is to create a well-defined compilation of resources and frameworks that support safe and effective use of AI in the practice of professional engineering.

In AI 101, you will find the following content

1. **Key Terms and Concepts:** This section will be a baseline overview of the key terms and concepts in AI, as well as some useful links to third-party resources.
2. **Application of AI:** An overview of the key differences between applying AI-enabled software and developing AI software.
3. **Opportunities of AI:** Examples of applications for AI and the types of problems it is being used to solve.
4. **Risks and Ethical Considerations:** A framework for identifying the unique implications and challenges that AI can introduce.
5. **Real World Examples:** Considerations for applying AI assistants, chatbots, and generative AI tools such as Microsoft Co-Pilot, Google Gemini, and Cohere North in a professional engineering setting.

Following this document, further work will be published to delve into applications of artificial intelligence in the practice of engineering, discuss ethics and risks of AI, and provide some frameworks for practicing engineers to assess how and when to apply AI in their day-to-day practice.

Key Terms and Concepts

What is Artificial Intelligence?

There is certainly no shortage of hype and enthusiasm around the field of artificial intelligence. The goal of this initial content is to cut through the hype and offer professional engineers practical insights into what is meant by AI. The hope here is to demystify the “black box” of AI by explaining at a high level the statistical techniques that underpin the field.

This is not intended as an exhaustive source of information. Links to third-party resources will be provided if the individual reader wishes to go deeper on the subject.

Artificial Intelligence: The field of AI is a multi-disciplinary field encompassing all the facets of replicating all or part of human intelligence. It entails the technical work of machine learning, the ethics of responsible application, and the specific problems that can be solved with AI.

Machine Learning, The Core Concept: “Replicating Human Intelligence”

This is the foundational goal and the philosophical bedrock of AI.

- **Artificial Narrow Intelligence (ANI)** is where AI is today. Systems are designed to replicate part of human intelligence to perform a specific task very well, often better than a human. Think of a language translation app, a medical image analysis tool, or a chess-playing engine. They are intelligent in a narrow domain but have no general awareness. For example, Large Language Models (LLMs) are statistical mimics of language that appear to reason but cannot.
- **Artificial General Intelligence (AGI)** is the future-facing goal of replicating *all of* human intelligence—the ability to reason, learn, and apply intelligence to solve any problem, just like a human being. It remains the ultimate horizon for many researchers in the field. To date, it remains an open question of whether or not this is achievable with our current technology

The “How”: “The Technical Work of Machine Learning.”

The discipline of **Machine Learning (ML)** is the engine driving modern AI. While AI is a broad field that historically included other methods (like logic-based systems), machine learning is the dominant and most successful approach today. ML is a subfield of AI where, instead of being explicitly programmed with rules, a system “learns” patterns and relationships directly from data. This is similar to engineers who use experimental results (data) to determine empirically derived physical relationships (patterns).

Analogy: The Reynolds Number

For example, consider the creation of the Reynolds number. In the mid-1800s, scientists **George Stokes** and **Osbourne Reynolds** collected data from experiments of fluid flow that they used to derive a formula to calculate the Reynolds number. This number identifies types of fluid flow (laminar, transition or turbulent). The process they used to develop this number is the same as the process of machine learning. Computers use vast amounts of data to recognize relationship patterns that exist across hundreds and even thousands of variables.

ML provides a scientific pathway to replicate the patterns humans recognize, from objects in a photo to patterns in written text.

Narrow Machine Learning

Narrow Machine Learning (NML) is a type of artificial intelligence trained to perform a specific and well-defined task. Think of it as a highly specialized tool designed for a single purpose, rather than a versatile, multi-purpose instrument. Popular techniques are regression, classification, and unsupervised machine learning.

Regression: Predicting Continuous Values

In ML, **regression** is a technique used to predict a continuous numerical value. The model learns from a dataset containing input features and their corresponding known output values. The goal is to find the underlying relationship between the inputs and the output to make accurate predictions on new, unseen data.

Practical applications of regression can include lifecycle prediction on mechanical parts, energy trend analysis and cost prediction.

Analogy: Stress-Strain Curve

In professional engineering, regression techniques are used to produce the stress-strain curve. This is a fundamental concept in materials science, where engineers plot the amount of stress applied to a material against the resulting strain (deformation).

- **The Data:** Just as a machine learning model is trained on data, engineers perform numerous tests on a material to generate data points for the measured stress applied and the amount of strain observed.
- **The Model:** The best-fit line or curve drawn through these data points is analogous to the regression model. This line doesn't just connect the dots; it represents the inherent relationship between stress and strain for that material.
- **The Prediction:** Once the curve is established, an engineer can use the model to predict the strain (a continuous value) for a given level of stress that wasn't explicitly tested. This is precisely what a regression model does: it predicts a continuous output based on new input.

Classification: Assigning to a Category

Classification is another machine learning method where, instead of predicting a continuous value, its goal is to assign an input to a specific category or class. The model learns from a dataset where each input is already labelled with its correct category or class. Note that when a ML algorithm determines a category or class, it is actually outputting the probability that it believes a given instance belongs to one class.

Typically, values over 50% are assigned in the positive, and values less than in the negative. However, this is a parameter that the programmer can choose. The important thing to remember is that a ML classification is a probabilistic assignment, and not a deterministic one. So, engineers evaluating classification predictions may want to ask for the raw probabilities so that they can determine which predictions are more or less confident.

Analogy: Soil Classification

A familiar concept for civil and geotechnical engineers is soil classification. Based on properties like particle size, moisture content, and plasticity, a soil sample is categorized into a distinct group (e.g., sand, clay, silt, or more specific classifications like "well-graded sand" or "high-plasticity clay").

- **The Data:** An engineer collects various soil samples and performs tests to determine their properties. Each sample is then manually classified. This labelled data is like the training data for a classification

model, such as the **Unified Soil Classification System**.

- **The Model:** The set of rules and criteria an engineer uses to decide the soil type is analogous to the classification model.
- **The Prediction:** When a new, unknown soil sample is found, the engineer applies these rules to classify it. A machine learning classification model does the same: given new input data, it predicts which predefined category it belongs to. For instance, a model could be trained to look at images of concrete and classify them as “cracked” or “not cracked.”

Unsupervised Machine Learning: Finding Hidden Patterns

Unlike regression and classification, **unsupervised machine learning** works with unlabeled data. The goal here is not to predict a known output, but rather to discover hidden patterns, structures, or relationships within the data itself. A common unsupervised technique is clustering, which groups similar data points.

Analogy: Modal Analysis

For structural and mechanical engineers, modal analysis provides a compelling analogy for unsupervised learning. In modal analysis, engineers study the natural frequencies and shapes at which a structure or component will vibrate when excited.

- **The Data:** Imagine having a large set of vibration measurements from a bridge or a machine part, without any prior knowledge of its structural properties. This is like having unlabeled data.
- **The Model and Discovery:** By analyzing this vibration data, engineers can identify the distinct “modes” of vibration—the specific frequencies and corresponding shapes at which the structure naturally resonates. This process of finding the inherent patterns in the raw data, without being told what to look for, is akin to unsupervised clustering. The algorithm groups the vibration data into clusters, with each cluster representing a different vibrational mode. This reveals the underlying dynamic characteristics of the structure.

In unsupervised learning, the goal is similar – take a large amount of data points and sort them into meaningful groups that can then be studied further. This technique is often used in marketing to segment different customer profiles based on their historical purchasing habits.

Data: The Raw Material for Machine Learning

Just as an engineer needs accurate material specifications and site measurements, a machine learning model needs high-quality data to learn from. This data comes in two main forms: structured and unstructured.

Structured Data

Structured data is highly organized and easily searchable. It fits neatly into rows and columns, like a spreadsheet or a database table. Each column represents a specific attribute, and each row represents a single observation or data point.

Analogy: Bill of Materials (BOM)

Think of a bill of materials for a large assembly or a structural analysis output table from a **Finite Element Analysis (FEA)** software. In a BOM, each row is a specific component, and the columns are clearly defined attributes like part number, quantity, material, and weight. It’s rigid, predictable, and easy for both a human and a computer to read and interpret. Other examples include sensor logs with timestamped readings, material property databases, and survey coordinate data.

Unstructured Data

Unstructured data is everything else. It has no predefined model or organizational structure, making it much more difficult to process. It's the messy, real-world information that doesn't fit into neat tables.

Analogy: Written Reports

Think of the written report that accompanies the structural analysis, the photographs from a site inspection, or the video feed from a drone surveying a construction site. These contain invaluable information, but it's not organized in a predictable format. Text documents, emails, images, audio files, and videos are all classic examples of unstructured data.

Vectorization: Turning Raw Materials into Usable Parts

Machine learning algorithms are mathematical at their core; they understand numbers, not pictures or words. Therefore, to use unstructured data, like images and text, it must be converted into a meaningful numerical format. This process is called **vectorization**, where we turn images, videos and text into a **vector** (an array of numbers).

How Images Become Vectors

An image is fundamentally a grid of pixels. Each pixel has a colour value. For a simple grayscale image, each pixel can be represented by a single number (e.g., 0 for black, 255 for white, and shades of gray in between). For a colour image, each pixel is often represented by three numbers for its **Red, Green, and Blue (RGB)** values.

To convert an image into a vector, we simply "unroll" this grid of pixel values into a single, long list of numbers. Imagine a small 10x10 pixel grayscale image. This gives us 100 pixels in total. We can create a vector of length 100 by taking the pixel value from the first row, then the second, and so on, and lining them all up. If it were a colour image, the vector would be 10x10x3 = 300 numbers long. In addition, there would be a value to store the original pixel dimensions of the images, so that it could be turned back into an image at another time.

How Text Becomes Vectors

Text is converted into vectors using techniques that capture its meaning and context. Here are two common methods:

1. **Term Frequency-Inverse Document Frequency (TF-IDF):** This method evaluates how important a word is to a document in a collection of documents. It gives a higher score to words that appear frequently in one document but rarely in others, making them good identifiers for that document's topic. Again, this is a probabilistic representation of how linked two words are to each other.
2. **Word Embeddings (like Word2Vec):** This is a more advanced technique. It represents words as dense vectors in a multi-dimensional space. The key idea is that words with similar meanings will have similar vectors; they will be "close" to each other in this space. For example, the vectors for "king" and "queen" would be closer to each other than the vectors for "king" and "concrete."

Analogy: Material Properties

Think of material properties. We can describe steel with a vector of numbers representing its density, yield strength, Young's modulus, and thermal conductivity: [7850 kg/m^3, 250 MPa, 200 GPa, 45 W/mK]. This vector numerically represents the "meaning" of steel in an engineering context. Similarly, we can describe aluminum with its own vector of properties. Word embeddings do the same for words; they create a numerical

representation (a vector) that captures the word's semantic "properties" and its relationships to other words.

Deep Learning

The Neural Network: A Simple Decision-Making Chain

A **neural network** is a computing system inspired by the biological brain. It's made of interconnected nodes, or "neurons," organized in layers. The most basic network has an input layer, one "hidden" layer, and an output layer. It is important to note that while this model was inspired by the human brain and the concept of interconnected neurons, these models fall short of replicating the human brain.

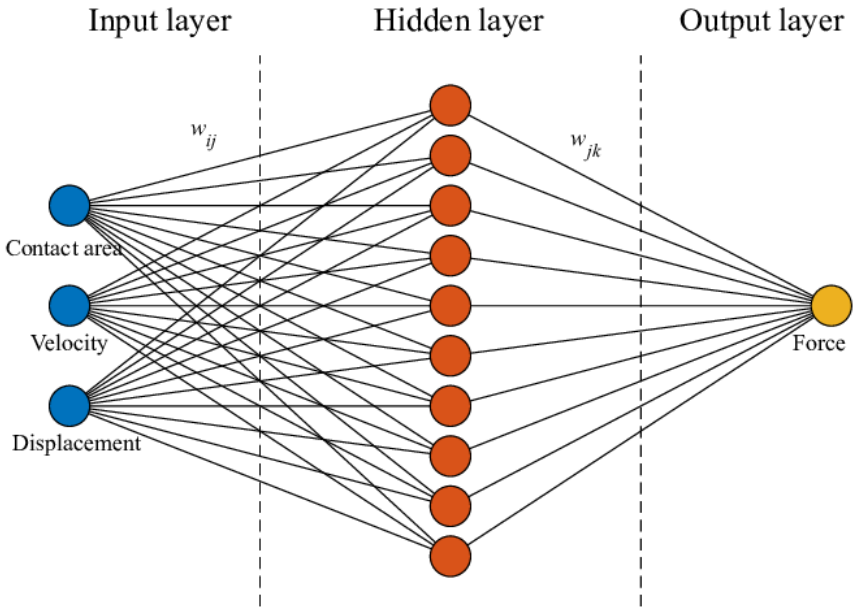


Figure 1

Data is fed into the input layer, and each neuron in the hidden layer makes a simple calculation. It weighs the importance of the inputs it receives and passes its results forward. The output layer then makes the final decision or prediction.

Analogy: Load Path Analysis

Think of a simple load path analysis in a basic truss structure. The external forces (the inputs) are applied to certain joints. The internal forces are then distributed through the members in the middle of the truss (the hidden layer). Each member carries a certain amount of tension or compression based on the geometry and loads. The final reactions at the support points are the output—the result of how those forces were distributed through the structure. The network "learns" by adjusting the "stiffness" (weights) of each member to achieve the desired output.

Deep Learning: Adding More Layers of Complexity

A **Deep Neural Network (DNN)**, the core of deep learning, is simply a neural network with many hidden layers—sometimes hundreds or even thousands. This depth allows the network to learn much more complex patterns by building on itself. Early layers learn simple features (like edges or colours in an image), and subsequent layers combine those features to recognize more intricate concepts (like shapes, objects, or faces).

Analogy: Building Frame

The progression from a simple truss to a complex, multi-story building frame is a great analogy for this concept. A single-story frame can analyze a simple load distribution. But in a 50-story skyscraper, the loads from the roof are transferred down through each successive floor. The columns on the 30th floor have to handle the cumulative load from the 20 floors above them. Each floor acts like a layer, processing the load from above and passing it down. A deep network works similarly, with each layer processing the output from the previous one to understand increasingly abstract and complex features.

The Problem with Deep Networks: The Vanishing Gradient

As these networks got deeper, a significant technical problem emerged: the **vanishing gradient problem**. During training, the network learns by making small adjustments to its neurons, starting from the output and working backward. This adjustment signal is called the "gradient." In very deep networks, this signal would get progressively weaker as it travelled back through the many layers. By the time it reached the earliest layers, it was so small (it had "vanished") that these crucial initial layers stopped learning effectively.

Analogy: Construction Manager Part 1

Imagine a construction manager on the ground floor of a 100-story building trying to give instructions to the crew on the roof by shouting. The message is passed up from floor to floor. By the time it reaches the top, the message is likely distorted, faint, or completely lost. The "gradient" of the voice vanishes with distance. The workers at the top can't act effectively because the instructions are too weak. Similarly, the early layers of a deep network couldn't learn because the error signal from the output was too faint.

Transformers: A New Way to Communicate

Transformers are a revolutionary type of neural network architecture introduced in 2017 that largely solved the vanishing gradient problem. Instead of a rigid, sequential, layer-by-layer processing, transformers use a mechanism called **attention**. Attention allows the network to weigh the importance of all input data points simultaneously, regardless of their position. It can create direct connections between any two points in the input sequence.

Analogy: Construction Manager Part 2

Instead of shouting instructions up 100 floors, the construction manager is given a **full set of blueprints and a radio**. Now, they can instantly see the entire project and communicate directly with the crew on the roof, the team on the 50th floor, or the workers on the 10th floor with equal clarity. The blueprints provide global context (the whole structure), and the radio provides direct connections (attention). Transformers work this way; every part of the input data (like every word in a sentence) is visible to every other part, allowing the model to understand context and long-range dependencies far more effectively.

The Foundation for Generative Models

Transformers enable deeper and deeper models to be trained and are the foundation that enables large language models.

This ability to understand the deep, contextual relationships between all parts of a dataset is what makes transformers the foundation for modern generative models (like ChatGPT or image generators).

A generative model's job is to create new data that looks like its training data. To do this, it needs to understand not just individual pieces of data but the underlying patterns, structure, and rules of how they fit

together.

By using the “attention” mechanism, a transformer can look at a sequence of words and understand the grammar, style, and semantic relationships. When asked to generate text, it doesn’t just pick the next most probable word; it uses its holistic understanding of the context to craft a coherent and relevant sentence. For images, it understands how pixels form textures, how textures form objects, and how objects relate to each other in a scene.

Therefore, the transformer’s ability to overcome the limitations of older deep learning architectures and grasp the global context of data is the key innovation that enables the powerful and creative capabilities of today’s generative AI.

Application of AI

The deployment of artificial intelligence by engineering professionals encompasses a spectrum of activities and professional roles. This report differentiates between the following categories:

Applying AI: Utilizing commercial off-the-shelf software as a tool for resolving domain-specific challenges and executing routine professional tasks.

Developing AI: Creating software, platforms, or tools—intended for internal deployment or for commercial sale to third parties—that integrate AI functionalities or capabilities.

A Hybrid: Employing partially customized applications and implementing fine-tuning adjustments to pre-existing software solutions.

Each of these engagement models necessitates distinct obligations pertaining to risk management and professional accountability, fundamentally requiring a comprehensive grasp of the probabilistic nature inherent in AI systems.

The Probabilistic Nature of AI

The maintenance of **Machine Learning (ML)** models and artificial intelligence solutions in a production environment fundamentally differs from the maintenance of traditional software systems. The behaviour of conventional software remains static unless explicitly modified, whereas ML systems are data-dependent and probabilistic. Consequently, the performance of ML systems can degrade even when the underlying code remains unaltered. ML models derive patterns from historical data; therefore, shifts in data sources, data quality, or real-world operating conditions can render models less accurate or unreliable over time. Thus, ML outputs are inherently probabilistic, not deterministic. Performance must be statistically quantified, and acceptable error thresholds must be meticulously defined and continuously monitored.

Given that many AI systems ultimately influence or automate critical decisions, maintenance protocols must ensure that outputs consistently align with established business objectives, ethical mandates, and regulatory requirements. Effective maintenance and monitoring, therefore, necessitate the implementation of continuous technical, organizational, and governance frameworks. This comprehensive approach is essential for organizations engaged in applying AI, developing AI, or managing hybrid AI implementations.

Applying AI

Certain existing software tools frequently utilized by practicing engineers, such as **Autodesk**, are beginning to incorporate AI-enabled features into their products. Similar to the adoption of any new software, the

implementation and application of AI software across an organization requires comprehensive planning to ensure it's used effectively.

This planning may include:

- The establishment of usage policies
- Staff training
- Execution of change management procedures
- Implementation of testing and quality assurance protocols
- Development of software updates and maintenance programs

Developing AI

The development of artificial intelligence encompasses the creation of AI software, ranging from individual components, such as a **Large Language Model (LLM)**, to fully integrated software designed for a specific task.

Specific tasks within this domain include:

- Data Engineering – The design, construction, and maintenance of systems responsible for the collection, storage, processing, and transformation of raw data into accessible, usable formats.
- Data Science – The application of statistical methodologies for the design and training of machine learning models.
- Software Development – The assembly and construction of AI platforms, systems, and products.
- Product Development – The process of comprehending and defining the specific problems that AI software should address and the optimal methods for resolution.

This type of work is frequently considered software development and generally falls outside the scope of what is defined as professional engineering practice.

Hybrid Cases and Low-Code Tools

These user-friendly interfaces abstract away the complexity of underlying machine learning models and extensive programming, enabling a broader range of individuals—even those without deep data science or software engineering backgrounds—to create highly effective, semi-customized applications. These applications are designed to leverage the core strengths of generative AI, such as natural language understanding, content generation, and complex reasoning, to accomplish specific, high-value tasks within an organization.

The utility of these low-code platforms is a vast and growing ecosystem of hybrid applications. These include:

Intelligent Knowledge Retrieval Systems: Applications that synthesize information from vast corporate document repositories, providing context-aware and natural language answers.

Automated Content Generation Tools: Customized platforms for drafting internal reports, marketing copy, or technical documentation that adhere to specific brand or style guidelines.

Advanced Data Analysis Assistants: Tools that allow users to query complex datasets using plain language, generating insights and visualizations without requiring Structured Query Language (SQL) or complex scripting.

Personalized Customer Service Agents: AI models tailored to a company's specific products and policies,

capable of handling sophisticated customer inquiries and escalating to human agents only when necessary.

In essence, the low-code generative AI movement represents a paradigm shift, moving AI from a niche, expert-driven technology into a widely accessible layer of organizational infrastructure, empowering employees to build bespoke solutions for almost any challenge.

Agentic Workflows

To fully grasp the capabilities unlocked by these low-code platforms, it is instructive to examine concrete architectural patterns. The concept of agentic workflows serves as a prime example. In this model, an AI system, acting as an "agent," is configured to autonomously orchestrate a sequence of steps, potentially interacting with various external tools and internal data sources, to achieve a defined objective. This contrasts sharply with traditional, single-prompt AI interactions by introducing multi-step planning, memory, and tool-use capabilities.

Robotic Process Automation (RPA) is a well-established methodology for automating repetitive, high-volume processes. A typical illustration is the processing of job applications, where applicants submit an online form containing their information and an attached resume, which is then routed and triaged to the responsible hiring manager for evaluation. Historically, workflow automation has depended on a predetermined set of rules. The system is susceptible to failure when confronted with novel or infrequent occurrences. Agentic workflows leverage AI to introduce flexibility into the system by managing more intricate and nuanced decisions within the automation. This results in a more robust system that offers enhanced utility and value.

There is a significant opportunity for agentic workflows to automate intricate, high-volume, repetitive, administrative duties that frequently consume a considerable portion of a professional engineer's time. Consequently, this area may serve as an initial entry point for many professional engineers utilizing AI tools. An illustrative example of an agentic workflow readily developed within a generative AI platform is provided below.

In this hypothetical scenario, the agentic workflow monitors a central repository for **Requests for Information (RFIs)** directed to engineers from active construction projects. It systematically analyzes the email, compiles pertinent project data, and forwards a comprehensive package to the designated engineer for a response. Figure 2 (page 14) outlines the various automated processes that this system could encompass.

This workflow can be conceptualized and implemented without requiring software developers, utilizing tools such as **Cohere North**, **Google Gemini**, or **Microsoft Co-Pilot**. However, a reflection on the automated decision-making processes depicted in the figure above should elicit several important considerations:

Permissions: Who is authorized to create these tools? What requisite training do they possess to ensure accurate implementation? Who is permitted to modify the logic within these tools post-completion? Who is granted access to this tool once it is deployed?

Risks: What are the potential impacts of an error? What is the projected frequency of such occurrences? How can errors be detected promptly? How do engineers validate the outputs daily?

Quality Assurance and Testing: Any agentic workflow necessitates rigorous testing. The development of a robust testing methodology may be intricate and must incorporate periodic re-testing of the solution over time. Who will be responsible for testing this tool prior to its utilization? Who must be consulted to establish the appropriate testing protocols for this tool? How will potential edge cases be identified? How will its performance be monitored once it is in production?

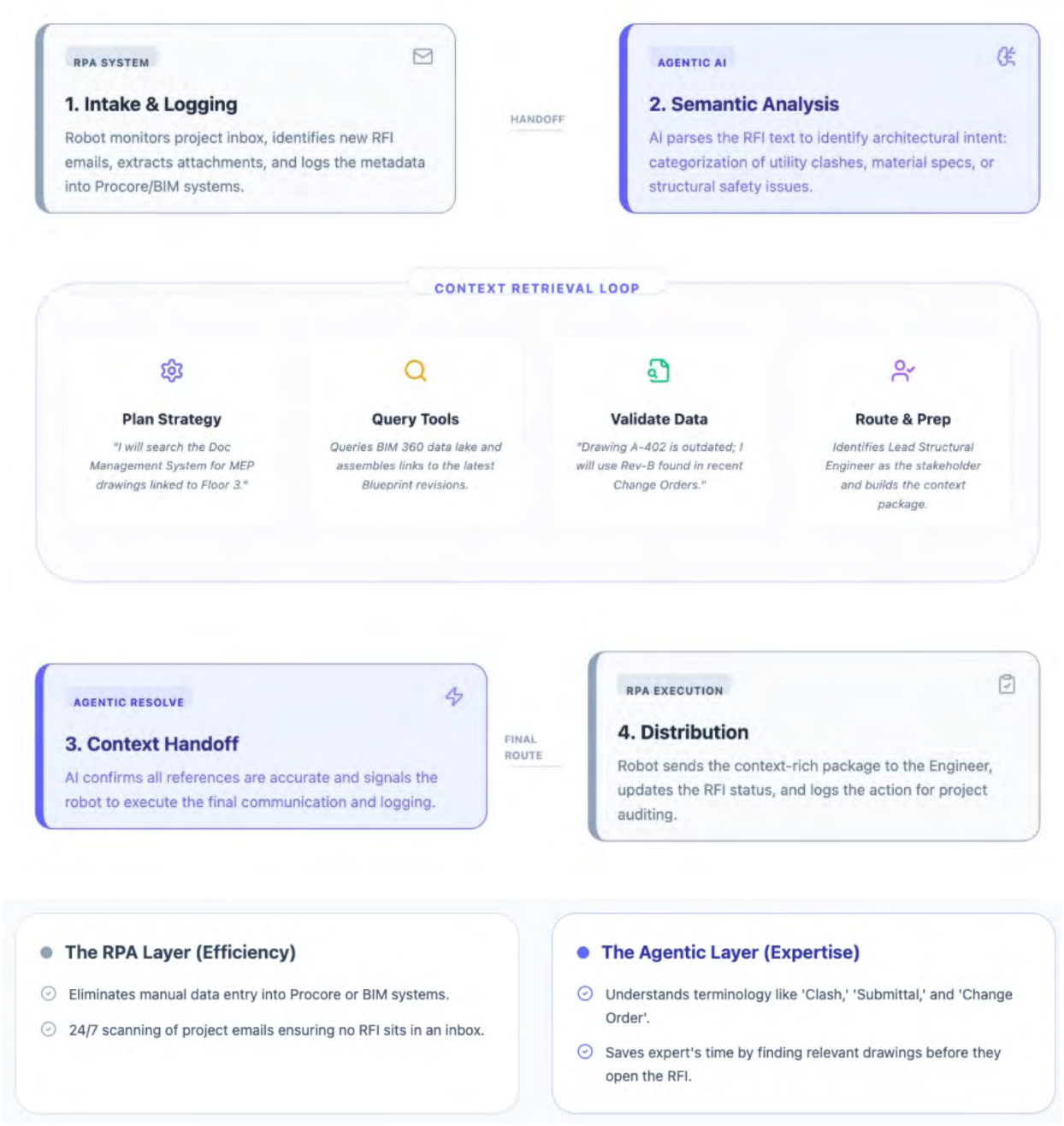


Figure 2

Maintenance: The ease of creation may result in an organization rapidly accumulating dozens or even hundreds of these operational workflows. The ongoing burden of regular maintenance can become substantial. Who will bear the responsibility for the sustained maintenance and enhancement of this tool?

This should impress upon an engineer that while low-code tools enable individuals without coding knowledge to automate a workflow, an experienced professional still needs to oversee the critical elements of responsible software development.

Fine-Tuning Large Language Models

Low-code tools democratize the complex process of fine-tuning large language models by providing intuitive, visual interfaces that replace thousands of lines of manual coding. This accessibility allows domain experts to specialize a general-purpose model on their own proprietary datasets, ensuring the AI understands unique industry jargon or specific organizational knowledge. By doing so, users achieve significantly higher accuracy

and more consistent brand alignment while often reducing the computational costs and latency associated with using oversized, generic models. However, this comes with significant risks that require data science expertise to navigate. Emergent misalignment is one such risk that engineers must pay attention to.

Misalignment occurs when fine-tuning a model on a narrow, seemingly harmless task or additional training data, unexpectedly and unintentionally, causes the model to exhibit broad, harmful or deceptive behaviours in entirely unrelated domains. It is “emergent” because these behaviours surface only after the specific training, rather than being explicitly programmed, and often appear in more capable, well-tested, larger models. Thus, unless engineers are also experts in the field of data science, they should exercise extreme caution in fine-tuning and or providing additional training data when deploying large language models as part of their workflow. This is of note as the fine-tuning of these models themselves is becoming increasingly accessible through low-code tools.

While low-code tools offer cost-effective accessibility, transitioning from experimentation to production-grade engineering requires the specialized expertise outlined in the AI Development section. Organizations should integrate internal upskilling or external support into their deployment strategy to ensure technical rigour and consistency. Without this structured approach, decentralized development by individual contributors risks sub-par performance, missed opportunities, and critical errors in engineering deliverables.

Conclusion and Discussion

Artificial intelligence is the technical work of machine learning, the ethics of responsible application, and the specific problems that can be solved with it.

Modern AI uses data to identify patterns and relationships, much like regression, classification systems, or experimental correlations that engineers already use. Today’s systems perform specific tasks but cannot reason or understand. Like any analytical or numerical method used in professional practice, the accuracy and usefulness depend on the quality of the data, the assumptions made, and how results are interpreted.

Recent advances, such as deep learning and transformer models, have made AI more powerful by allowing it to work with large, complex unstructured datasets like text, images, and video. This enables useful applications in design support, inspection, forecasting, and analysis. However, these systems are still mathematical models that produce statistical outputs, not independent thinkers.

While there are lots of concepts presented in this section, the intent is to instill one key message, which is that all AI, from regression to advanced large language models, learn complex patterns from a mathematical representation of the real world. This means that underpinning every single AI model is probabilistic statistics. Hence, fundamentally, AI outputs are probabilistic in nature and not deterministic. This does not mean that the outputs of AI are not useful, just that the level of uncertainty needs to be understood and accounted for by engineers using AI-powered tools.

For professional engineers, a purely technical view of AI is incomplete. As AI systems become more powerful and integrated into society, the ethical dimension is a core component of the field itself.

To build a complete understanding of Artificial Intelligence, consult *AI 101 – Part 2* to explore the ethics of responsible application, critical questions, and applied AI.

Appendix

Sources for Deeper Reading

The following freely available sources are recommended should a deeper understanding of the content be desired.

https://www.deeplearning.ai/courses/ai-for-everyone/

https://www.deeplearning.ai/courses/generative-ai-for-everyone/

https://cohere.com/llmu

Prominent Quotes

Below are some statements that you might have heard on AI:

Sundar Pichai, **CEO** of **Alphabet** and **Google**: Pichai has spoken about the profound and responsible integration of AI into Google's products and society.

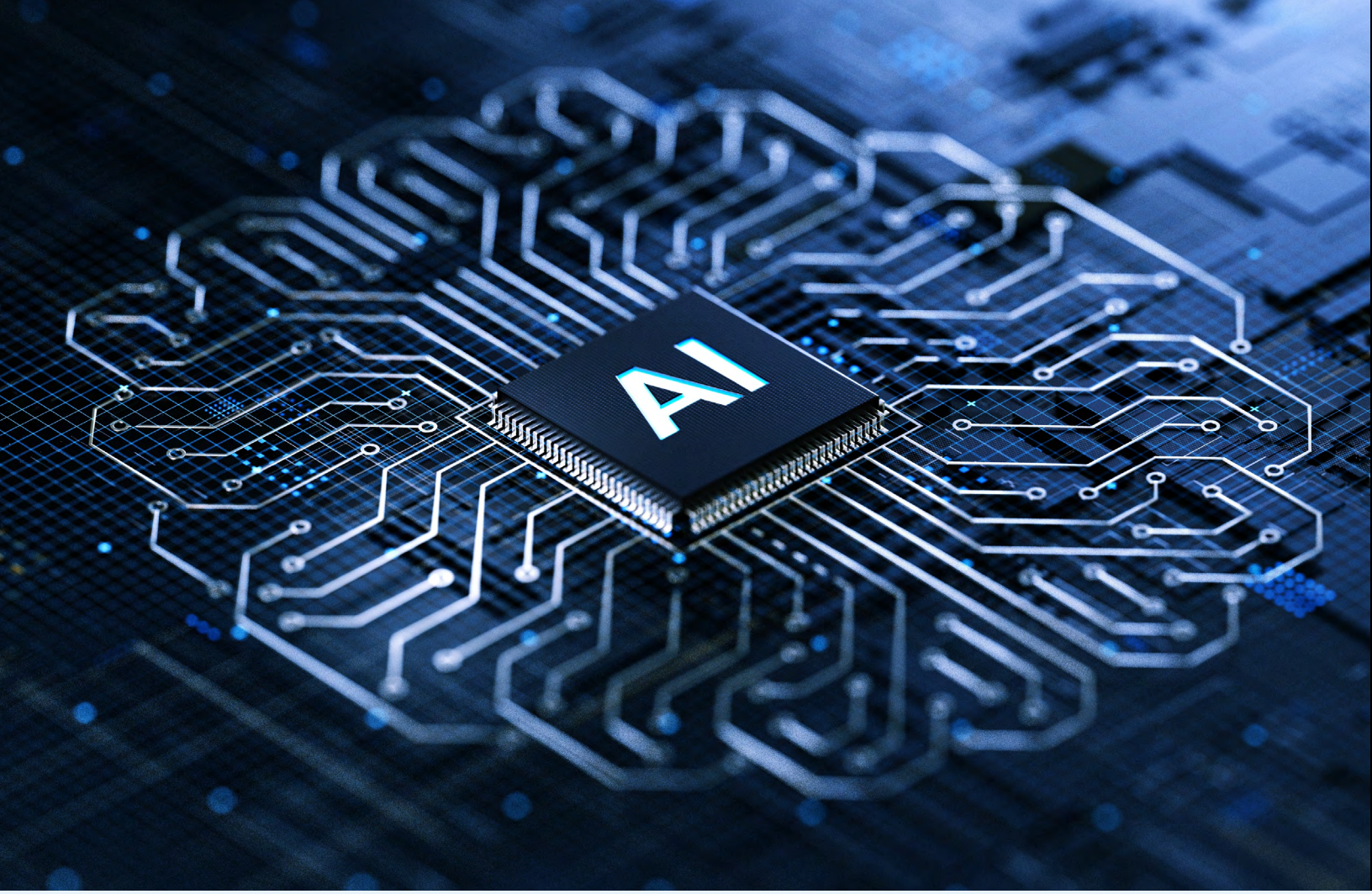
> *"AI is the most profound technology we are working on today. It has the potential to be more transformative than fire or electricity."* - **Sundar Pichai**, World of Datanomics, October 26, 2023

Jensen Huang, **CEO** of **NVIDIA**: As a leader in the hardware that powers much of the AI revolution, Huang has a unique perspective on its future.

> *"We're at the beginning of a new industrial revolution. And this is the production of intelligence. And the production of intelligence will be done in these new factories called AI data centers."* - **Jensen Huang**, NVIDIA GTC 2024, March 18, 2024

References

Figure 1: *Enhanced grip force estimation in robotic surgery: A sparrow search algorithm-optimized backpropagation neural network approach - Scientific Figure on ResearchGate*. Available from: https://www.researchgate.net/figure/The-basic-structure-of-the-neural-network_fig1_379403654 [accessed 18 Dec 2025]



Contact Us

Ontario Society of Professional Engineers
5000 Yonge Street, Suite 701
North York, ON, M2N 7E9
1-866-763-1654
info@ospe.on.ca

ONTARIO SOCIETY OF PROFESSIONAL ENGINEERS