Where this thing’s at			
Stage	Status	I’ll have this done by	Responsible
Brief	Done		
Outline	Done		
First draft	Done	Jun 5, 2024	Sam
Second draft	Done	Jun 10, 2024	Denys
Proofread	Done	Jul 3, 2024	Donnique/Caroline
First design draft	Done		Amanda
Second design draft	Done		Amanda/Donnique
Complete	Client please review		Donnique/Connor

Notes and resources

DRAFT

By Denys Linkov

AI automation on a budget: Getting started with high ROI use cases

Alternates:
AI automation on a shoestring: Budgeting and ROI without breaking the bank
From lab coats to Iron Man suits: Navigating AI projects on a budget
The true cost of AI agents, beyond the numbers
The true cost of AI automation: How-to budget and measure ROI
Your Guide to Building AI Miracles on a Budget

If your AI project feels like building an Iron Man suit out of scraps—you're not alone. Right now, everyone wants teams to spin AI miracles out of dust and dreams. But we can't all be Tony Stark, the genius AI wizard.

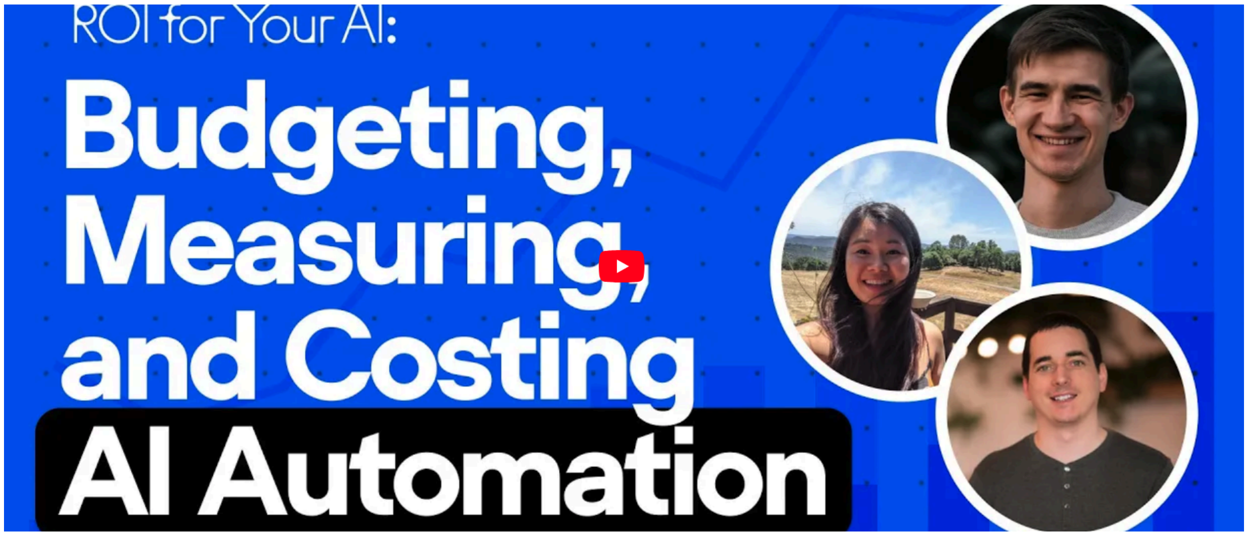
POV: Your AI team is the poor soul in the lab coat :(

Thankfully, most AI teams aren't trapped in the desert attempting to create customer support chatbots with rusty metal. But with budget constraints, the risks of deploying large language models (LLMs), and the technical burden on your stack, producing AI agents can feel like you're fighting for everything you need to succeed.

So let's take a step back and analyze the costs of building, deploying, and expanding effective AI agents. This will not be a listicle on the costs of each line item—although, if you'd like to see something like that, let me know. Instead, we'll be talking about how to approach strategizing and budgeting the time and resources necessary to build a sustainable AI operation.

We'll explore the strategy and costs associated with scaling agents across new use cases, how to weigh the cost of tools and technology like GPUs, choosing the right models (and why you might be choosing wrong), and how to budget for innovation and experimentation.

If you're more of an audio-visual learner, we suggest tuning into our webinar on this topic (featuring Colin Guilfoyle, VP for Customer Support at Trilogy, who managed to automate over 60% of their customer support in under two months). Stick around for the full breakdown below.



Starting small before scaling agents and use cases

Starting with a simple agent is the surefire way to gain traction, before scaling to numerous agents and use cases. Because when you try to walk before you crawl or run before you walk, you're going to trip over your own feet.

[Starting a new AI agent? We've got you]

But running before you walk isn't as common an issue for most teams. Most companies struggle to move their first AI proof of concept (POC) out of production purgatory—it's often called the cold start problem, named for the difficulty in starting old internal combustion engines when the gas is cold. Once the gas is hot, turning the engine on and off is a breeze. Similarly, once teams have launched one AI agent, they find it much faster and easier to expand to several agents or several use cases.

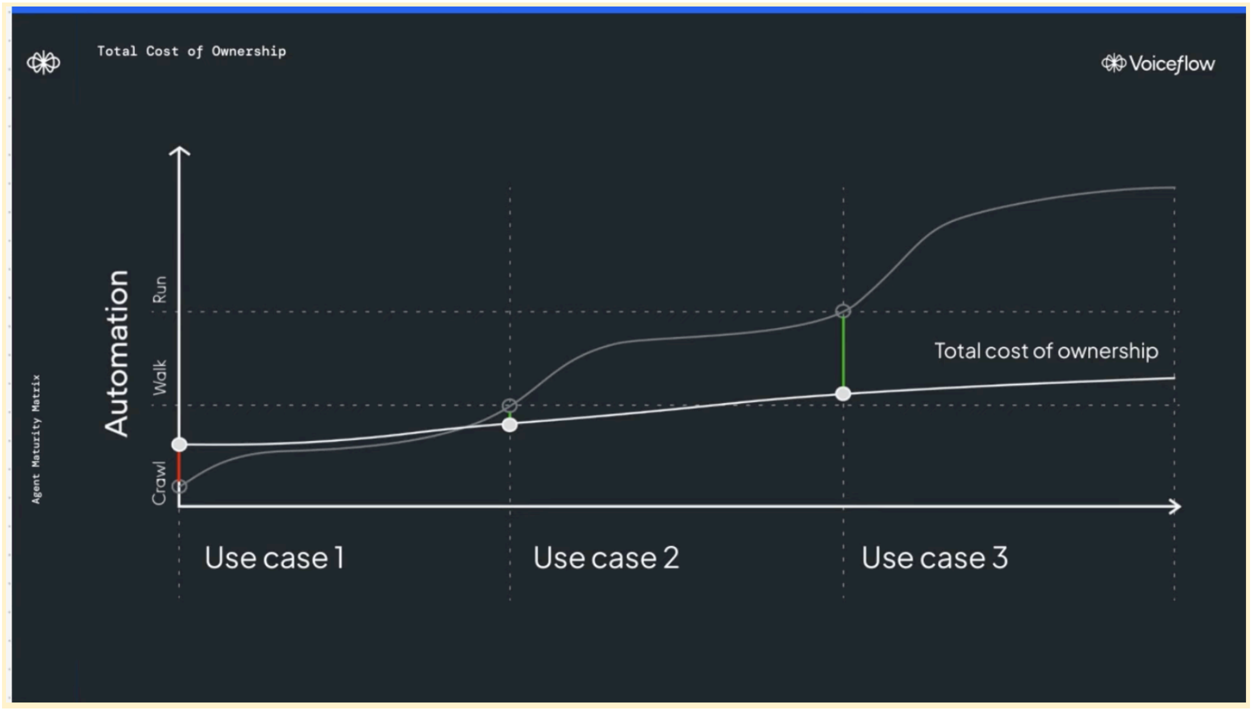
But you don't have to take my word for it. Colin Guilfoyle, VP for Customer Support at Trilogy, has done it. His team started with one AI build—they call it the Atlas core, very Tony Stark-coded—and have used that build to expand to 90 customer support lines that handle AI support across products.

In order to scale production to this level and keep all their agents working smoothly once they got there, they needed to start with how they organized their team. Because there's so many product lines that require support, their team is made up of code-focused, senior product specialists. These directly responsible individuals (DRIs) are given a subset of products to analyze each week, including how well the customer support automations and tickets are performing. Then, they replicate what goes right and refine what goes wrong—from refining knowledge base searches, training models, ticket raising, building the right retrieval-augmented generation (RAG), and integrating the right tools to solve specific product issues. They apply their best practices to their Atlas core, which they use as a foundation for building and expanding to new agents and products, and the process continues.

By effectively dividing and replicating their agents, while continually monitoring and improving them, Trilogy is on track to support 65% of their customer inquiries using AI agents. The next phase of their expansion includes replacing human support on L2 troubleshooting and automating customer changes in the system securely.

[Read more about how Trilogy automated 60% of their customer support in 12 weeks.]

While you may not have the budget to expand your team or want to create 90 agents, Trilogy's approach to iterative improvement and replication is a wise one to consider when weighing costs. When it comes to scaling your agent, whether that's launching more agents or expanding your agent's current capabilities, there's a lot you can do. Start with your minimal viable product (aka your single use case AI agent) and slowly layer in the use cases that expand its problem-solving. You'll know it's time to add new use cases when you've mastered the one you're currently on. In fact, you'll find the cost of ownership remains quite sustainable if you've built processes and integrated tools that are growing according to your needs.



As you scale, you'll experience savings costs when it comes to human support hours. Trilogy reduced their human support hours by 60% after 12 weeks, freeing support staff to focus their efforts more efficiently. In the long run, scaling sustainably allows your support needs to grow as you do while saving your team the one resource they can't get back—time.

“How much should I budget for this?”

People always ask me this question. And the answer is an unsatisfactory one—it depends. In the case of Trilogy, a large-scale enterprise that streamlines operations and support for hundreds of clients, they use a variety of tools ranging in cost, including:

- **Amazon Web Services:** For compute, hosting, backups, etc. They also use Lambas, the first port of call for tickets, which categorizes the issue and offers a response based on a knowledge base
- **LLMs:**
  - OpenAI, to generate responses
  - Enki, to cross-check LLM responses and choose the best one. If there isn’t a viable answer, Enki kicks the response back up a step to generate a better one
  - Anthropic
  - Occasionally, Gemini
- **Zendesk:** for managing and routing tickets
- **Voiceflow:** to design, produce, and launch AI agents

Large companies with AI support across channels can spend over $100,000 on LLMs, tokens, and associated AI costs. Similarly, it can cost up to the same per year on Amazon Web Services. And that doesn’t include the cost of engineering a sophisticated support system that automatically generates and cross-checks AI responses across 90 agents.

When you’re talking about tooling, people, and time, it’s hard to make estimates about how much you should spend on AI agents unless we talk through the minute details of your circumstances. (Shameless plug for my colleague Peter Isaacs, who would be stoked to talk through your AI automation journey in painstaking detail.)

My advice is to talk to your technology and tooling vendors, ask colleagues in your field, and do a lot of research. We’ve also included a RAG cost estimation template for you to forecast costs for your next project.

~~To get you started in your research, here are a few examples of tools you might run into while growing in your AI maturity, and how those costs might add up.~~

~~Estimated budget for tools and technology:~~

~~Remember when we said this wouldn't be a listicle filled with numbers? Well, we couldn’t resist giving you a little extra. Below, this chart shows some of the technology and tools you can use at each stage of your AI maturity, and what you can expect to budget for each. Keep in mind that if one technology or tool works for you in the crawl stage, you should try scaling that tool as you grow (before adding a new one to your tech stack).~~

~~**Stage of AI maturity**~~	~~**Technology**~~	~~**Estimated budget**~~	~~**Tools**~~	~~**Estimated budget**~~
~~**Crawl**~~	~~RAG~~		~~CustomGPT~~ ~~GPT Builder~~ ~~Microsoft Copilot~~ ~~OpenAI Assistants API~~	
	~~Context Windows~~			
	~~Memory~~			
~~**Walk**~~	~~*You may use the technology in the crawl stage, and include:*~~ ~~Knowledge Base~~		~~*You may use the tools in the crawl stage, and include:*~~ ~~LangChain~~ ~~Voiceflow~~ ~~Dialogflow~~ ~~Rasa~~	
~~**Run**~~	~~*You may use the technology in the crawl and walk stage, and include:*~~ ~~API Calls~~		~~*You may use the tools in the crawl and walk stage, and include:*~~ ~~IBM Watson~~ ~~Kore.AI~~	

5 tips for choosing your LLM models (spoiler: versions are underrated)

The number of LLMs available has exploded in the last year. The influx of choices brings questions about which ones you should be using based on your use cases. There are five things you can do right now to understand models and choose the right ones.

1. **Rely on RAG:** Build a robust knowledge base and use RAG to pull that information into your LLM when it's generating responses. This enhances the effectiveness of your LLM by providing useful context for your responses from your datasets, documentation, and FAQs. Don't underestimate how powerful RAG can be. The more context your AI agent has, the fewer LLM calls it needs to generate a relevant response, the more cost-effective your agent can be.
2. **Model versions matter:** Many people complain about OpenAI's GPT, claiming that it's getting worse with each new version—but it's unlikely OpenAI is releasing worse versions of their flagship product. What's happening is that the newest version no longer works for your particular use case. Don't trust the AI leaderboard. Spend time on prompt engineering. Do multiple tests before you land on the model and version for your use

case. For many projects, using an older LLM version will offer results on par with the newest version and be more budget-friendly.

3. **Use models to cross-check responses:** As previously mentioned, Trilogy layers their LLMs atop one another. This tiered approach would begin with your agent using an NLU to match your user's response to an intent you've already mapped, like collecting account information or surfacing a help link. If your agent can't find a match, it moves down the order priority and uses RAG to search your knowledge base and find sections of documents you've uploaded that have the closest semantic similarity. If it finds a match, it'll use your LLM to generate an answer to address your user's intent. Then cross-check that response with a different LLM, generating two responses and choosing the best one. This process has multiple benefits, including better quality control, more accurate AI responses, reduction in hallucinations, more concise responses, and improved data collection.
4. **Test different models for different use cases:** Using a tiered approach can also help you test which models work best for your use cases. If you find that one model consistently "wins" at the quality control cross-check, it might be worth investing in that LLM over the other. For classification tasks, some use cases are better suited to GPT-4, but Haiku, one of the cheapest models, also performs well and should not be discounted. The newest version of Claude may not work as well for your support tasks as the previous one. The key is to test, evaluate, and iterate as you work with different models and versions.
5. **Weigh the cost of prompt engineering vs. upgrading your model:** This is where teams need to make decisions on accuracy, development costs, and runtime costs. You can put a massive context window into GPT-4 or Claude Sonnet 3.5 and you'd be spending a couple of dollars per interaction. You could also use smaller models but you'd need a way to measure the tradeoff—the cost of running the model compared to the business gains of increased latency. This is where having good evaluations is important. Improving the prompts also takes time for both the prompt engineering and surrounding systems. You have to make a large number of LLM calls to actually see a return on investment. You have to weigh how much time that prompt engineering and evaluation is worth. You might increase your LLM costs by upgrading your LLM, but that might be worth it if you've already optimized your prompts.

Choosing the right LLMs requires thoughtful intention. Use RAG to provide context, making whichever LLM you choose more efficient and cost-effective. Different model versions work better for different tasks, so don't be afraid to use older versions and cross-check responses to ensure quality and accuracy. You should be balancing the costs of prompt engineering against your models to help you achieve the best performance within your budget.

To GPU or not to GPU? That is the question.

A graphics processing unit (or GPU) has made the modern world of AI possible. Compared to CPUs, GPUs have many smaller processing cores designed to work in parallel. As a result, LLMs and other Gen AI models use GPUs to perform massive mathematical and operational tasks quickly and simultaneously. Today, enterprise, consumer-grade GPUs serve multiple uses, from model building and low-level testing to deep learning operations, like biometric recognition.

We won't go into all of the GPUs out there, because there are a bunch. But they are typically divided into three categories useful for enterprise:

1. **Consumer-grade GPUs:** Typically sold for gaming, but have been used for local model training and deployment, particularly open source models.
2. **Cloud-based GPUs:** Many cloud providers let you rent GPUs ranging from entry-level (T4s) to state-of-the-art clusters (H100s). A great place to get started when experimenting, training, or running models.
3. **Datacenter GPU clusters:** For larger companies, procuring your own GPU cluster or server becomes an option. These can be just the hardware installed in a data center or platform offering to get started faster.

The question is, do you need one? GPUs are expensive resources. For many, using proprietary models and a serverless approach gets them far enough in their AI journey to solve for most use cases. But for the folks interested in AI innovation and playing with bigger, faster, complex AI projects, a GPU has been a critical asset, leading to some supply challenges.

Choosing the right hardware for a use case is essential. It's overkill to build a cluster of H100 GPUs to run a seven billion model inference. It takes a lot of engineering hours to host a model, optimize inference, batch queries, and put up guardrails to make it run efficiently. Instead of investing in a GPU—and spending months installing and deploying models-—my advice is to leave it to platforms until use cases and costs are better defined. When you're building a large-scale AI operation, hiring a team to run innovation makes sense. But for most use cases, avoid the complexity and use CPUs and smaller models more often. Bigger isn't always better.

Add research, eval-driven development, and experimentation to your budget

The conversation around AI seems to center around avoiding risk and not getting left behind. It's a pretty negative approach to an exciting and novel technology, and that affects how we evaluate the value of AI and budget for it. But a budget represents more than just money, it represents time, effort, and strategic thinking. Instead of thinking about all the ways things can go wrong, invite your teams (and even your leaders) to budget for:

- **Keeping up with AI:** Budget the time necessary for your team to understand the AI landscape. Colin's team at Trilogy spends two hours a week on Twitter, LinkedIn, and Reddit, learning, engaging with new information, and expanding their AI knowledge. Because of this, they're proactive about addressing new use cases and experimenting with tools. When executives come to them with requests, they're ready to respond, either with a plan to adopt new ideas or an explanation of their previous experiments. Budgeting time for AI makes their team more productive, knowledgeable, and adaptable to change.

- **Evaluation-driven development**: AI projects aren't always clear on returns, but that hasn't stopped every company under the sun from adopting some form of AI technology. So, if we've already accepted that, it would serve us to evaluate the ROI of AI accordingly. Budget your engineering prowess behind evaluation-driven development (EDD), a methodology for guiding the development of LLM-backed projects using a set of task-specific evaluations, like expected prompts, contexts, and outputs as references. These evaluations guide prompt engineering, model selection, and fine-tuning to help you quickly measure improvements or regressions as your project changes. Don't just measure how many tickets you automate. Determine what parameters you'd evaluate success in, and work backwards.
- **Experimentation and known problems:** Finally, you need a budget to experiment and roll out new tools, tech, and use cases. There needs to be support from leadership for this. AI moves quickly and if your AI team is keeping up with the changes, they'll also need a budget to experiment and react to those changes. On the other hand, don't let shiny new tools and ideas have you too focused on problems under the streetlight instead of known issues AI could solve.

It's not too late to invest in the time, evaluation, and experimentation you need to succeed with AI. The most important problems aren't the easiest ones to solve, but an organization that is forward-thinking about AI will see ROI faster than a reactive one.

Let's build Iron Man-level AI on a start-up budget

Remember, you don't need to be Tony Stark to achieve results with AI. By starting small and scaling up, carefully budgeting for tools and technology, and prioritizing continuous learning and experimentation, you can make the most of your budget, no matter the size.

RESEARCH

Jun 4, 2024

What is a GPU?

GPUs were originally designed primarily to quickly generate and display complex 3D scenes and objects, such as those involved in video games and computer-aided design software. Modern GPUs also handle tasks such as decompressing video streams.
The "brain" of most computers is a chip called a central processing unit (CPU). CPUs can be used to generate graphical scenes and decompress videos, but they are typically far slower and less efficient on these tasks compared to GPUs. CPUs are better suited for general computation tasks, such as word processing and browsing web pages.

How are GPUs different from CPUs?

A typical modern CPU is made up of between 8 and 16 "cores", each of which can process complex tasks in a sequential manner.
GPUs, on the other hand, have thousands of relatively small cores, which are designed to all work at the same time ("in parallel") to achieve fast overall processing. This makes them well suited for tasks that require a large number of simple operations which can be done at the same time, rather than one after another.

Read more: Demand for computer chips fuelled by AI could reshape global politics and security

Traditional GPUs come in two main flavours.
First, there are standalone chips, which often come in add-on cards for large desktop computers. Second are GPUs combined with a CPU in the same chip package, which are often found in laptops and game consoles such as the PlayStation 5. In both cases, the CPU controls what the GPU does.

Why are GPUs so useful for AI?

It turns out GPUs can be repurposed to do more than generate graphical scenes.
Many of the machine learning techniques behind artificial intelligence (AI), such as deep neural networks, rely heavily on various forms of "matrix multiplication".
This is a mathematical operation where very large sets of numbers are multiplied and summed together. These operations are well suited to parallel processing, and hence can be performed very quickly by GPUs.

https://theconversation.com/what-is-a-gpu-an-expert-explains-the-chips-powering-the-ai-boom-and-why-theyre-worth-trillions-224637

GPU Technology Options for Deep Learning

When incorporating GPUs into your deep learning implementations, there are a variety of options, although NVIDIA dominates the market. Within these options, you can choose from consumer-grade GPUs, data center GPUs, and managed workstations.

Consumer-Grade GPUs

Consumer GPUs are not appropriate for large-scale deep learning projects, but can offer an entry point for implementations. These GPUs enable you to supplement existing systems cheaply and can be useful for model building or low-level testing.

- NVIDIA Titan V—depending on the edition, this GPU provides between 12GB and 32GB of memory and between 110 and 125 teraflops of performance. It includes Tensor Cores and uses NVIDIA's Volta technology.
- NVIDIA Titan RTX—provides 24GB memory and 130 teraflops of performance. It includes Tensor and RT Core technologies and is based on NVIDIA's Turing GPU architecture.
- NVIDIA GeForce RTX 2080 Ti—provides 11Gb memory and 120 teraflops of performance. It is designed for gaming enthusiasts rather than professional use and is also based on NVIDIA's Turing GPU architecture.

Data Center GPUs

Data center GPUs are the standard for production deep learning implementations. These GPUs are designed for large-scale projects and can provide enterprise-grade performance.

- NVIDIA A100—provides 40GB memory and 624 teraflops of performance. It is designed for HPC, data analytics, and machine learning and includes multi-instance GPU (MIG) technology for massive scaling.
- NVIDIA v100—provides up to 32Gb memory and 149 teraflops of performance. It is based on NVIDIA Volta technology and was designed for high performance computing (HPC), machine learning, and deep learning.
- NVIDIA Tesla P100—provides 16GB memory and 21 teraflops performance. It is designed for HPC and machine learning and is based on the Pascal architecture.
- NVIDIA Tesla K80—provides up to 24GB memory and 8.73 teraflops of performance. It is designed for data analytics and scientific computing and is based on the Kepler architecture.
- Google tensor processing units (TPUs)—while Google TPUs are not GPUs, they provide an alternative to NVIDIA GPUs which are commonly used for deep learning workloads. TPUs are cloud-based or chip-based application-specific integrated circuits (ASIC) designed for deep learning workloads. TPUs were developed specifically for the Google

Cloud Platform and for use with TensorFlow. Each provides 128GB memory and 420 teraflops of performance.

DGX Servers

NVIDIA DGX servers are enterprise-grade, full-stack solutions. These systems are designed specifically for machine learning and deep learning operations. Systems are plug-n-play, and you can deploy on bare metal servers or in containers.

- DGX-1—provides two Intel Xeon CPUs and up to eight V100 Tensor Cores, each with 32GB memory. It is based on the Ubuntu Linux Host OS. DGX-1 includes the CUDA toolkit, NVIDIA's Deep Learning SDK, the Docker Engine Utility, and the DIGITS deep learning training application.
- DGX-2—provides two Xeon Platinum CPUs and 16 V100 Tensor Core GPUs, each with 32GB memory. It provides significant scalability and parallelism and is based on the NVSwitch networking fabric for 195x faster training than the DGX-1.
- DGX A100—provides two 64-core AMD CPUs and eight A100 GPUs, each with 320GB memory for five petaflops of performance. It is designed for machine learning training, inference, and analytics and is fully-optimized for CUDA-X. You can combine multiple DGX A100 units to create a super cluster.

Learn more in our guide to NVIDIA deep learning GPU, which explains how to choose the right GPU for your deep learning projects.

Top Metrics for Evaluating Your Deep Learning GPU Performance

GPUs are expensive resources that you need to optimize for a sustainable ROI. However, many deep learning projects utilize only 10-30% of their GPU resources, often due to inefficient allocation. To ensure that you are using your GPU investments efficiently, you should monitor and apply the following metrics.

GPU utilization

GPU utilization metrics measure the percentage of time your GPU kernels are running (i.e. your GPU utilization). You can use these metrics to determine your GPU capacity requirements and identify bottlenecks in your pipelines. You can access this metric with NVIDIA's system management interface (NVIDIA-smi).

If you find that you are underusing resources, you may be able to distribute processes more effectively. In contrast, maximum utilization means you may benefit from adding GPUs to your operations.

GPU memory access and usage

GPU memory access and usage metrics measure the percentage of time that a GPU's memory controller is in use. This includes both read and write operations. You can use these metrics to optimize the batch size for your training and gauge the efficiency of your deep learning program. You can access a comprehensive list of memory metrics through the NVIDIA-smi.

Power usage and temperatures

Power usage and temperature metrics enable you to measure how hard your system is working and can help you predict and control power consumption. These metrics are typically measured at the power supply unit and include resources used by compute and memory units, and cooling

elements. These metrics are important because excessive temperatures can cause thermal throttling, which slows compute processes, or damage hardware.
Time to solution
Time to solution is a holistic metric that lets you define a desired accuracy level, and see how long it takes you to train your model to reach that level of accuracy. That training time will be different for different GPUs, depending on the model, distribution strategy and dataset you are running. Once you choose a GPU setup, you can use a time to solution measurement to tune batch sizes or leverage mixed-precision optimization, to improve performance.

https://www.run.ai/guides/gpu-deep-learning

ROI for Your AI: Budgeting, Costing, and Measuring AI Automation

https://youtu.be/k6rnd_SGNQ4?feature=shared

Denys:
Hey, it's Denys from Voiceflow. In case you missed it, we have the recording for ROI for Your AI with myself, Chip and Colin. Hope you enjoy, and let's dive into it.
So thank you everybody for joining. I'm Denys, I lead our machine learning team here at Voiceflow. I've been with the company for the past three years, so I have seen the adventures of GenAI unfold. Excited to welcome Chip and Colin to our panel today. I'll hand it over first to Chip, and then to Colin, to introduce themselves.

Chip:
Hi, my name is Chip. I'm VP of AI and Open Source at Voltron Data. So we work on building GPU native query engines and we also contribute to a lot of open source project including Apache Arrow, Ibis and Substrait. Very excited to be here today.

Colin:
My name's Colin Guilfoyle. I'm the SVP of customer service in Trilogy, which is a private equity company. We have about 90 products under our management, and so I manage the CS team providing CS services on those 90 products, and then some other AI stuff as well as we've kind of gotten more into being AI first as a company.

Denys:
Thank you for introducing yourselves. Let's check out who's here. So we had some intros in the chat here, but we also have some pretty interesting companies and sectors joining us or attendees from these companies. So I think this just shows how much people think about the cost of AI budgeting and ROI. We have healthcare, banking, core tech, insurance, telecom, so a

lot of really, really interesting conversations I think that we'll have today. And we'll try to touch on examples from different industries.
So for our agenda today, we'll go through four key pillars here. In the pillars, we have our... We'll talk about the first project and experimentation, how to get started with GenAI and budget that first project. Then we'll focus on the growth side, so growth in hiring. Then we saw quite a few questions from the group about model choices in RAG. We'll finish off with the budgeting side on hardware, so compute and GPU and finally rounded up with some big takeaways.
So for these kinds of events, we love to make sure that we cover the topics you're interested in. So quite a few list of interesting topics here that the group has submitted. We put those out under these different categories and we'll make sure to touch upon each of them. So to get started with the first project component here, maybe I'll pass it over to Colin to get started. So maybe talk a little bit about your first GenAI project that you worked on and managed and tell us a little bit about that.

Colin:
Sure. So we in Trilogy went with AI fairly early on, but the first major project that went to production was actually with yourselves. Because we manage 90 products and we're trying to be kind of standardized, but at the same time to do a first initial concept, we picked one of our carrier product, which is a firewall, and we decided to transition from the provider we had, which was Forethought, which was just a kind of fairly basic AI assisted knowledge based search, and we went to trying to actually replace agents and time on tickets and fully resolve tickets within chat. And we did that via Voiceflow. We are big on data and we had a lot of information there, so we took the tickets for the last chat and tickets for the last I think three months, aggregated out what the commonalities were, and basically once we were able to address 30% of the tickets fully by AI, we went live.
Ironically, the first intended one was to use it for voice, but we very quickly pivoted and deployed it via chat. And we only recently actually had deployed voice because the chat kept us so busy. But yeah, that was kind of the first project. It was strange. We tend to deploy very quickly and then iterate even quicker. We found that no amount of testing of an AI in a local environment was kind of enough. We had to actually deploy it and get customer interactions because it was only then, as with anything, that you actually saw what people were doing and what they were interacting.
We were testing going, "Here's the problem I'd have," and we were being very specific because most of us are engineers, so we were like, "Here's the logs, here's the error I'm getting." And what happened was when we deployed it in the wild, people would just say computer broke and figure out the rest from that. And so it was very much the first project was iterated out, but the results were super-fast and kind of found. We deployed I think in three products in the first two weeks, and we went to all 90 products within, I think it was two months.

Denys:
I think that's a pretty interesting metric of trying to get 30% automation before going live. And I think a lot of us have felt that in the machine learning space, you have your training data, you sort of validate prototype, and then you go live and then what your customers say or what real world data looks like is completely different. So I think it's an important journey that all of us face

and sometimes we forget it. It's almost like you do one project and you hope that next time the data will be better, but it's not. So really important item to keep in mind.
I guess over to you, Chip. You started a company a couple of years ago, sold it, joined Voltron. Maybe you can share a little bit about your experience of running a startup project, having this idea and how you budgeted and tried to build the startup. So I think there's a lot of people in the audience interested about that.

Chip:
Hey, yeah, so I think I'm pretty lucky that I think I get to see AI application as different level. So at one level I do advise a few startups, so I do get to get involved. So of course it is advising, sometimes you don't get very hands-on, but sometimes I try to be very hands-on with certain projects that I find very interesting. So I get to see them developing this from my prototyping to productions, getting feedback. And a couple of the companies that I'm pretty involved with actually recently launched their project. And in a way similar to Colin, it seems like I'm personally am very interested in beyond just techs. So I see that a lot of people using LLMs for text base, but they do think that the future is multimodal. So I'm especially focusing on applications when not just the input is multimodal. So for example, on projects that I'm actually helping a company with is trying to convert from screenshot to code. So it's quite challenging the evaluations. Another, I'm so interested in outputs of this multimodal.
So for chat, right, you just use a text conversations ability, but if you want to do a 3D, you have to do voice, visual, facial expressions, action generations. And this has been a big, big challenge because one thing is that not quite a lot of evaluations benchmarks for those. If I look for things like role-playing benchmarks, there are a few coming mostly from China. I think that China is getting it. I don't know why, but I think on the benchmark I fight on role-playing is from China. So I think on the aspect of our application side. The other is I also help a few companies putting platforms, like fine-tuning platform. So I do test out a platform for them. So I'm experimenting with how to fine-tune a project. And I think it's a budget in case of time, how much time we have for this, but also money. Do we have data, how much money we can allocate to data creation, how much money to like API calls, and then how much money for evaluations because if we do AI as a judge, we can be quite costly as well. So yeah, that's where I am.

Denys:
Yeah, for sure. I think that that cost of trying to estimate how much the project costs for LLMs is something that everybody thinks about. Whether you're running evaluation data sets, synthetic data generation, something that's really important. We can probably talk about that a little bit more in our growth stage since launching the first project, you might just want to pick a cheap model, pick Haiku or ChatGPT or Lama, launch that project and then see what works. But I think different companies have different risk thresholds. So in Colin's case, automate 30%, let's get into production. Other companies want to do a lot more evaluation. So think a lot of great points there for getting started. Maybe let's move on to the second component here, talking about the growth stage. So transitioning into that growth stage of iterating through products, improving your existing ones.

Within Voiceflow, we try to take this framework into account. We call this the crawl walk, run framework. Not super unique, but in the case of conversational AI, conversational design, the general idea here is that you want to start with something simple. Start with one assistant. In the case of Colin, I think it was three assistants, get that working as a minimal viable product and then slowly layer things on and scale. And if you do that right at the beginning, you'll actually find that your cost of ownership will remain fairly flat because you've built out the patterns, built out the tools to do so. I think that's something that's really important. And on the team side, people know how to build an additional agent, the best practices. You have playbooks to follow. Very interesting area is once you go from zero to one, it's a question of one to five. And even after that, if you have 90 assistants, I guess managing that is quite different as well. So I guess maybe on that business growth topic, heading back to you Colin, how do you manage 90 assistants in production? What does that look like?

Colin:
The way our teams are built, obviously we have the SVP and VP team, and then we have what we call our conductor team, which are very code heavy senior project product guys. And so we kind of made them what we call DRIs, directly responsible individuals. So we gave them each a subset of the products to analyze each week the results coming in. Voiceflow and chat is one element of it. At this stage we've kind of stacked a lot on top of that. So we just call our system Atlas at this point. And so basically Atlas is a squeeze of tooling and automations that allow us to automate both tickets and chat interactions so we can reuse both. So with the DRIs, they were working on automations and AI kind of training and building the right RAGs and the right tools to allow us to solve more and more of that specific products issues.
And then we have some smaller products as well. All 90 are not massive beasts, some are much kind of smaller, maybe end of life products. So for those, we had one DRI who's responsible for basically keeping it at a base level that it can do the basics right. The KB searches, the [inaudible 00:12:03] detection, the ticket raising, what we call the L2 troubleshooting. So basically troubleshooting using past tickets as well as knowledge base and access to the code and logs. So basically that's how we kind of kept it together. We had DRIs that would be responsible for specific subset and then VPs sitting above them who would be responsible for groups of those. But we have about... We're kind of separated into BU0s. So different types of products go to different BUs. And so the VPs would be responsible for maybe two BUs, two use cases, and then the DRIs would be responsible underneath them for the individual metrics, making sure everything is being worked on.
I think for us, initially the goal was 30%. I think by the month three we were at 50%. We're currently trending towards 65%. Either this week or next week we'll hit our quarterly goal a month early, which is nice. But that iterative gain from here is tougher. We got the low-hanging fruit with the 30%, we got the slightly higher fruit, I guess it is, for the last couple of months. And now it's very, very detailed. We're trying to replace L2 troubleshooting, which is exceptionally difficult. We're trying to automate making changes in the system securely so that when a customer is looking for something to be

done, that it can be done that way. So yeah, that's kind of how we've grown it. Is to make sure there's always somebody iteratively improving it each week and tracking that improvement.

Denys:
That was very detailed. Really appreciate that structure of the team. I think a number of people are wondering about that is how do you structure that team. So that EBP, SVP, then BP, by business unit, and then every sort of product manager under that responsible for project. I think that's a very powerful concept for teams that are scaling and growing. I guess over to you Chip, about figuring out how to know when to hire somebody else. **So in the case of Colin's org, there's people already there, it's a reasonably sized team. In a startup you don't always have a lot of money and you want to prioritize things. How do you do that?** How do you figure-

Chip:
Hey, so I'm not sure if it's me because there was a little bit of a cutout. So I think my understanding is that the question is how do I decide when to hire? I think the question is pretty much the same, whether it's AI, not AI, we hire when we realize the lack of this role is causing us, it's not having this growth. It's important natural growth. So maybe one question specifically, just recently the committee came to me and said, "Hey, should we bring our labeling in-house?" Because they had been sourcing the labeling out of house and it's very, very expensive. So it was looking like, okay, depending. So it is a question of whether how much labeling needs do you have in the futures and also how much time it takes for you to run maps in-house and also this return investment for that look like. And also understanding the challenges of building an internal labeling in-house.
So people think of labeling, it's just like, okay, here is a guideline, here's the labeling. But especially for a lot of AI, especially from complex stuff, it can be very hard to create a guideline, and then hiring qualified people to do that. So it's a lot of operational challenges. So if you want to bring that in-house, you have to take on the operational challenge you have to manage and hire the team. So yeah, so I think we just hire when it's an issue of talent needed. Also, another team, this just came to me, do we want to bring research in-house? I think a lot of AI companies want to build their own model. It's like, okay, "We want to build this AI research lab."
And I usually want people to not have an in-house research team because it's actually very expensive to maintain. And I know that a lot of people want to publish papers like it's a conferences, but it's quite a lot of work by publishing papers. Actually a discussion we had recently, it's like, "Hey, we know this conference is great and internationally we have had a lot of success hiring from the conference, so let's try to write a paper and publish in this high profile conference and have us hire." But the thing is that you can spend a lot of time writing a paper and you don't know for sure that paper wouldn't get accepted. So yes, before we think about bringing on hiring researchers, we just need to think about our own. What we want to get out from it and what's the cost?

Denys:

Those are some really great points. I think on the data labeling side, I think yesterday there is a tweet about the CTO of Databricks still spending 15 minutes a day labeling data. So that's pretty interesting is that the value of having good labeled data is so high that some of the most senior folks in companies are still spending time on it, but for a long tail of certain data, there's certainly a high operational cost of maintaining that. And at that point it's a question of are your data sets worthwhile to create in-house or buy them or pay a company externally to do that? Because for specific domain use cases, data can become a bottleneck. Touching on the second part about research teams, I think that that's super interesting. I think at the beginning when large language models were sort of going more mainstream, the first model came out, Databricks was training model, Bloomberg was training models. I think a couple more, JPMC. That was seen as a path forward, but a lot of companies right now are just fine-tuning a lot of open source models.
So I think that that's a good approach and there's a lot more services now that let you do that if you have well-labeled data. So I think for a lot of people who are thinking about fine-tuning their own models, I think it's a great approach and better than training from scratch, but still a lot of challenges there and maybe beyond the scope of this conversation. But really important things to think about as you scale and figure out making that decision between your own models and external models.
So I think to wrap up this topic to Colin's earlier point, talking about some of Trilogy's growth. So giving that 90 set of product lines and getting that automation up to 60, sounding like 65% now. Pretty interesting milestone there, just what's possible. And I think getting that last 20% is going to be quite the challenge because people are saying interesting things to your chatbot or to your assistant and you need to know how to manage that. I guess Colin going back to you about the tools that you're using. **So obviously using Voiceflow, but maybe can you break down some other tools that you're using and a rough cost estimate per set of tools?**

Colin:
Sure. Costing tools is a bit difficult because I mean we're pretty much in an AWS environment and spread... We're a tech company, so we're spread across many different pools of resources. In terms of other tools, so Atlas is a core structure, most of it... So our ticketing system is Zendesk, that shouldn't be taken as an endorsement for their amazing AI abilities. It's just our ticketing system which has an API on it that we use to put tickets through. Basically using Lambdas through AWS, we code in based on triggers. So for the chat, obviously it comes in via Voiceflow. For tickets themselves, they go into a Lambda, which basically its first port of call is what we call Atlas ticket. That has a knowledge base of the types of problems that we solve and it has a knowledge base of what we need to solve those problems.
And so what it does first is it categorizes the issue, kind of checks to see if we have enough information because again, of those customers just saying computer broken, and it iterates through those, gets the information we need from the customer, extracts it, and then tries to solve it. And then for solving-wise, we have at the moment basically built two L2 troubleshooting bots. One is a kind of chain prompt that is using pass tickets, KBAs, log analysis as I said. It goes very, very deep. The L2 bot on that side can take anything up to I think 12 minutes to fully go through its process and come back with

an answer. We have a second L2 bot, which is more streamlined. Again, it has access to the tickets and the knowledge base, but it doesn't do much else, but it's quicker.
And so we pull those two answers together. And then we have another AI who we've called Enki, and Enki reads the two L2 submissions, picks which one is better, if neither of them are good enough, it then sends it for an agent to work. So everything's based around Lambda's, some databases, and basically there's obviously an AI-first database in the background with all the ticket information from the past. That's something we found was quite a learning curve. So we were always very good at documenting and tickets as a process because we're a 24-7 operation, anyone can be picking up a ticket, but there was always a lot of information in there that was only necessary for a human agent. And so one of the things we've had to do, is spend a lot of time doing, is stripping out that information that was really useful for human but is absolutely confusing for an AI.
In our earlier days, it was coming back when an answer that was basically just a summary of an internal process that somebody followed to close the ticket, not the actual how the ticket was closed. So yeah, from our perspective, we spend easily, I think the spend is well over 100K on AWS a year, considerably more I think. And we use OpenAI, we use Anthropic, we don't really use Gemini, but it is there, but we would spend again, well over 100K on OpenAI alone per year in terms of tokens and stuff. So it's quite substantial spend there. We obviously spend a few dollars on Voiceflow as well. And other than that, we also have an experimentation budget, which we kind of bring in products just to test them. We'll take them for either a year or a couple of months, see if they're useful, see if they can do the job. If they can't, we just move on to the next one. But that's kind of where the budget mostly goes. I'd say of the two highest things would be AWS and OpenAI.

Denys:
Yeah, thanks for sharing that. I think really important things to call out is that you need multiple services to build your solution, especially when you're in production. Things can be quite pricey, but even combining all those costs together, there's still obviously some cost savings there. I think the problem you brought up in terms of large language models and humans needing different information is super interesting. I think we found something similar is that certain pieces of context are fairly quote-unquote, "called distracting" to a large language model. I guess Chip, you've done a decent amount of human evaluation, LLM evaluation. Have you seen any similar kinds of patterns?

Chip:
When you say human evaluations patterns, like the cost patterns, or what do you mean?

Denys:
So you had that blog post about the LLM arena evaluations and people picking certain prompts versus LLM-based evaluations. Have you seen any similar patterns between how LLMs evaluate versus how people evaluate responses?

Chip:
Yeah, so actually I was about to ask Colin about that. Colin, you mention you have an AI to pick out a better response of the two models. One is stronger, one is weaker. Why do you need that?

Denys:
So obviously we have our customers as in the customers of the products, but we also have the BU kind of customers as well. And we need to basically prove that what we're sending is accurate. So we would always have had a QC model where somebody would read it and verify it. And so basically we implemented the same. As well because we have two completely different LLM models, as in not models as in just the LLM, but how it troubleshoots is completely different. So for our perspective, it's really useful to have data as to which one is providing more accurate answers. Ironically, what we found up until recently at least, is that the simpler one was giving better answers. And so for us it's to focus those DRIs who are responsible for those L2 bots troubleshooting wise, that they would make good decisions.
And as well, for example, as I said, with the data that's in the tickets, if we're getting poor quality data in, it would be... So all our products are kind of fairly technical. We're talking telco products and stuff that are used by large corporates. So for us to make a mistake and send something that might not be great out is not acceptable. So the kind of Enki is our last line of defense to make sure that whatever's being sent out is at least useful.

Chip:
But Enki's AI, so basically you're trusting AI to evaluate and it seems-

Colin:
Yeah.

Chip:
... like from... Yeah. Interesting. So in the simple model is better, maybe most of them?

Colin:
Has been, yeah. I'm not going to put pressure on the guy who is responsible for the more complex one. He's getting slightly more correct and it is improving. But from our perspective, it was just interesting. We gave it all this extra tooling, it costs a lot more to run per iteration and it wasn't delivering that much more. And so it made that we had to focus and go, "Okay, why isn't it and how can we fix it?" So that's kind of where it is.

Chip:
Yeah, so I think using AI to evaluate, I see it's very common nowadays. **It's probably the fastest growing evaluation method in production. Mostly there's no other way to do it, right? I think in LinkedIn recently they have a blog post and they just have humans every day annotate a 500, and that is their NordSTAR metrics, to see if their models are improving. But other than that, usually you just have to rely on AI. And I see this AI also**

has a lot of biases, so ways that humans do. So one thing is better clearer guidelines. I don't think AI judges will work. So I felt like it's seen a lot of problem because AI judges, it really point on what model is being used and what promise is being used. And I just went through a bunch of public, you have on this type evaluation tools and they provide this faithfulness is coherence, and they have the prompts for those. And I went through those prompts was just yesterday, and they're all very different and none of them are very good. Some of them are extremely verbal.
So I like, oh my god, if we use this style of prompts, it would cost us a lot of money. So I think it's like you just have very clear guideline. If humans disagree on the guideline, I don't think AI would go to pick up other. I think they say some weird things. For example, they say, oh, AI is very bad at numerical scores. So say that people have somehow conversion like okay, one, two, three, four, five, it's using the guy's score that AI can do reasonably well, but AI has been empirically been pretty bad continuous score. Maybe when you ask, hey, between zero and one to output the value, it's pretty bad at that.
So actually I saw [inaudible 00:28:30] example. When it says like, hey, you can ask AI to put the confidence score. And I was like, huh, why is [inaudible 00:28:38] showing the example? Because [inaudible 00:28:41] AI it is not good at outputting from zero to one. So I don't know. So maybe there's some special... If people have tried that, I would love to see if people have found consistent numerical scores from AI. There's some different kind of biases where human's judges and AI judges have shown to be different. So for example, humans have a residency bias, so we remember what is last. We tend to give it high score, but we have felt that AI tend to prefer the first, what was the first option is usually get the highest score.
But at the same time it's been different with Llama 3. Like we have found the Llama 3 information with the bottom somehow. I think it just saw someone tweet like last week, so I haven't validate it yet. So it seems like it could change with the models as well, like residency bias, the top of the bottom of the prompts is taken into account more. So I think AI suggestion is a very interesting approach and still a lot of work to be done to understand how to use well.

Denys:
Yeah, a lot of great insights there and details. I think bringing that back to the ROI side, I think it's really important for people to understand what's your budget per evaluation, because you can put a massive context window into GPT-4 or Claude Opus, or one of the expensive and powerful models, and you'd be spending a couple of dollars per interaction, which isn't ideal. I think this is a really important topic to be aware of, is that improving your prompt might improve performance by some cases 1%, some cases five, some cases 10, sometimes negative. At Voiceflow we've done some research on that. But sometimes it works.
I think I was reading about there was a medical benchmark data set and some folks used DSPY to optimize a prompt. DSPY is a framework for optimizing generating prompts and they found that a 6,000 token prompt was most effective, but that would be very expensive for some of these models. So really important to figure out how you're evaluating your models, which models you're using, how big are your prompts, how

many examples are you using? It's I think an active area of research and applied research and something that we'll continue to see. But from a business perspective, I think just knowing to ask that question is what is one percentage point of accuracy worth? And it'll definitely depend on the use case.

Okay, so this goes nicely into model choice and RAG. **So I guess in this area we want to focus a little bit about what kind of models have people used before, which ones work for different use cases, and maybe talking a little bit about the economics of RAG. So I know Colin, Trilogy uses RAG quite consistently. What kind of insights have you found with your agents and assistants?**

Colin:
For us, we've had mixed results. How do we split the data out and stuff has been the most complex thing to try and at least reliably search. From our perspective, what has been clear is it moves and it changes. So we've had really good results from a RAG that we built with a model of... For example, we created a RAG on some of our code base and then took in some other programming documentations from an engineering side and then we were checking to see answers. And for whatever model we used at first, I think it would've been one of the four turbos, we got great results, we were getting great answers and it very quickly just went off on a tangent and was no longer useful to us. So from that perspective, our focus is more on the models than the RAGs that we're using to search. At this stage we just kind of take the tickets in as much as we can.
Once we have an index that we can search, we pull the ticket in as much as we can in full so that we have the full context of what we're doing. Model-wise, when Claude 3 came out, we found weirdly that the middle model, which I can never remember the name of, but basically the most advanced model, it wasn't giving us the information we wanted. It was terrible to be honest. Whereas the second, the middle model was actually much more robust and giving more accurate answers. So yeah, from us it's still where the model is predominantly for turbo, the latest one, and for some of the Atlas ticket work, still the classification is easier and better than kind of 3.5, the lower models. It seems to be more accurate. It's not as creative, which maybe is what you need at that stage of the puzzle. So oftentimes it'll be different models called along the process. But yeah, from our perspective, for the most part we'll just use whatever's the newest and move with it because generally speaking it will give better results.

Denys:
It's actually very interesting to hear. So we have Opus is the most expensive Claude 3 model.

Colin:
There you go.

Denys:

Colin's been using Sonnet, which is the middle one and ChatGPT-3.5. I think with some of the work I've been doing with classification task with large language models, we've actually found that GPT-4 generally is quite good. But there are cases with when ChatGPT outperforms it. I think one benchmark I ran, Haiku did the best, which is the cheapest model. So very interesting area to continue to figure out and as part of that it gets pretty challenging to manage all these models and prompts.

So I think as teams scale, also figuring out how to manage that. I guess Chip on your end, models, RAG, what kind of data experiments have you done to figure out what's best?

Chip:
So I think this brings me to questions because I've seen a lot of people complaining, it's like, oh hey GPT-4 is getting worse over time and time. And then it's like okay, imagine you're OpenAI, would you intentionally put out models that's just worse over time? Of course minus this thing about safety alignments, you don't want to process bad responses like censorship whatsoever, right? Usually I wonder if this a questions of... OpenAI has a lot of different use cases than GPT-4. So maybe once they evaluate they just evaluate on a set, I don't know how many tests sets or how many tasks they evaluate on and just launch the best model overall. So that means that some of the tasks, the model might be really a lot better, but some other tasks, maybe the model is getting worse.
So maybe we hear a lot of people complaining about GPT-4 getting worse could just be the people who are on the subset of the tasks, because if a models performing well, we pretty don't hear people complaining. So I just think, so maybe this even highlights the importance of you just can't just take on the leaderboard or on this reported supposedly best models at face value. We just need to do a lot of time evaluating these models. And also I see another thing when evaluating models is that sometimes it could just be maybe you're just not good at prompt engineering. I've seen for some other use case, I try. For some use case I could try for weeks, I still couldn't get really good performance but for some I was just like, oh, I'll just change the prompts a little bit, and the performance go from unusable to really pretty good. So sometimes I'm not sure when a company says, oh this model sucks for us, and we want to know how good are the prompts? How good are the data you're fitting into the system? So it's really tricky.

Denys:
Yeah, for sure. I think it's an ongoing problem I think. Had attendee share their experience with Claude as well being quite well and highlighting the importance of different syntax as well. So I think that this topic will continue to evolve. As we go through, people still figuring out which models are best to use and the challenge again there to Colin's point as well, that the versions really matter. So it's not just the model is that which version of the model that you're using.

All right, another topic, I think we'll pass this over to you Chip, **about compute and GPU. I know you've done quite a lot of work on this. One of the products that you were working on sort of showcased the importance of using GPU for a lot of big data handling. Maybe you can share a little bit about that. And also just for AI use cases, how do you deal with GPU costs, how do you estimate them, how do you choose a specific GPU type? It would be great to get your insights there.**

Chip:
So yeah, thanks for Denys for showing this. So this is a benchmark that we Voltron Data released recently. So basically it showed that at a certain data scale when you move from Spark to TCS, so Spark here would run on CPU and TCS would run on GPU. We can see this the cost and the runtime both significantly lower. So I think it's one of one things about GPUs is that I think GPUs are just going to be here to stay. I don't think that go back to less powerful computers. Historically I have never seen that happen. So a lot of people today use GPUs for training and inference, but I also see this a big category of workloads, people are just not using GPU for, which is data processing like ETL work. And up until very recently there was a huge GPU shortage and some people was like, okay, "We don't have any GPUs for training inference. Who has GPUs for doing other stuff?"
But I'm not sure how people here think about it. But I think that might change or it's already changing. I'm not sure anyone here has tried to acquire a lot of GPUs recently. I think it's pretty funny in the last couple of years you can see startups, Pagedeck, literally is a competitive advantage. It's just like they have access to GPUs. And I think I was talking with the VCs recently and they were asking me what point do you think that having access to GPU would stop being a competitive advantage? I feel like coming from Nvidia, I'm very bullish on GPUs, but I do also think that the production of GPUs is also pretty fast, ramping up pretty fast. So I don't think access to GPUs is going to be as much of a problem that it was maybe in just a year ago. We see a lot of people providing GPUs for cheap. Anyway. Sorry, what's the question?

Denys:
Just talking about choosing a right GPU for a use case. Does it even make sense to use a GPU? When does it make sense? Or should people keep calling proprietary models and sort of using a serverless approach there?

Chip:
I think it's like GPUs, I think it really depends on what models you run. If it's smaller models, you have a lot of options. So I think the idea is very important is what people call a hardware luxury. So some models get popular not just because the models, because they fit the hardware and hardware GPU today use 16 gigabyte of RAM, 24 gigabyte of RAM, like 80 gigabyte of RAM. You can see that the model size in the both parameters just making just enough to run on certain hardware. So I think smaller model you have a lot of choices, but you have bigger model, you just need to shower money for bigger machines. And one question people usually ask that I hear a lot is like if you're training

and I hear that, but inference, is there any point to get much more powerful machines to run inference?
For example, H-100, right? People say okay, it's probably an overkill to use H-100 to run a certain 7 billion model inference. But so it really depends on how good can you at optimizing the model inference, can you batch queries? I don't think it makes sense you just use a H-100 and just serve one query at a time, but you can batch queries. And if you can do it more efficiently then maybe it's worth it to go with bigger models. So I think it's another point of people ask, "Hey, should I host model or shouldn't use an inference service?" It's just a lot of work getting into hosting a model and optimize inference service, getting it in streaming, batching, doing guardrails or yeah, it's just a lot of work like making inference service run efficiently. So maybe that could be something some teams should keep in mind if they think about hosting their own models.

Denys:
Yeah, for sure. I think there's a few questions about that and people often forget about how much it costs to pay people. So if you have an engineer working on optimizing a model, that'll cost you at least $1,000 a day probably if you're in the U.S. And the question is how many API requests can you deal with that? I think my perspective on the topic is generally try to use a third party hosted solution.0 It's somebody's business that they focus on. Right now the rates are pretty cheap actually. They've fallen I think over 600 times since GPT-3 was out with that initial rate. So for most scales, I think third party makes sense. If you get to a certain scale, then hiring your own team and running it will probably start to make sense. But for most people in the room and the use cases we're talking about, try to avoid that complexity.
And if you do want to run your models many times, you can use smaller models. Something we do at a Voiceflow and we actually run them on CPUs. So yeah, not always necessary to get the biggest in Chinese GPUs.

Yeah, third party APIs do quite well. So have around 12 minutes remaining. So maybe from each of our speakers, maybe just a summary, something you want the audience to take away and then we'll dive into Q&A. I'll start off with Colin. Sort of on this topic of budgeting, AI costs, et cetera, what's a word of wisdom you'd give to our audience?

Colin:
On budgeting? For us the important thing has been having a budget to just test and roll out. So to actually have something where we can... And we're lucky I suppose, the atmosphere in Trilogy has been from the very start AI first. So we have a lot of backup from those on high and those who kind of approve budgets, that we need to be spending and investing in AI. And what we've found kind of twofold, one, we invest in the actual people who work here. So everybody is obliged almost to spend two and a half hours a week minimum on Twitter, on LinkedIn, on Reddit, learning and evaluating, kind of upscaling in AI. And we found that that has been immensely helpful and that has a budgeting impact. Obviously that's like two and a half hours they're not working tickets. And when I say working tickets, that's just for the L1's and L2's. This is all the way up the top where I'm the same, everybody above me is the same.

So I think actually investing a bit of time and making sure that the people are kind of getting up to speed in AI and adapting to it and then having a budget to, as I said, to experiment. So as I've said already at the start, we use Voiceflow. It's probably one of the reasons I'm here. We deployed it, but we still evaluate competitors to Voiceflow pretty much every other week.
For two reasons. One, to see what's out there and maybe push yourself or Derek an email and go, "Why doesn't yours do this?" Or because we know we're going to get questioned on it from up high. So in other words, oftentimes we'll get a message saying, "Hey, I saw this on Twitter." I can give an example I suppose. We get a lot of posts like Intercom, "Why are you using Zendesk when you could be using Intercom?" And we are able to say, "Well actually we've evaluated it, we can give you a long list of reasons why we don't use it." And so having that ability, that kind of scope there to go out, experiment, buy new software if we need it or rent it or whatever, but just to be able to push each time and test. So from our perspective, it's having an experimentation budget and having a budget to actually let our guys get up to speed and the time to get up to speed.

Chip:
I think AI is interesting. As instance, a lot of company I've seen that's enterprise that don't want to invest in things unless it knows their returns. But sometimes for emerging technologies, maybe the returns is not quite clear. And sometimes if it's a forced enterprise, companies just take a little bit more risk, be willing to. Because another risk, actually an interesting thing that I saw, I'm not sure which company did that research, but 7% of companies chose AI because there was existential risk because they believe if they don't choose AI, they're going to die because they're going to be replaced by another company that uses AI. So I do think that, yeah, it's not quite clear the returns, but sometimes it just need to make some judgment on whether it's worth investing in. Also, what people call is evaluations-driven development.
So in software engineering we have a test-driven development, right? When you write the test cases first before you write a code, and I think that for AI applications we just need to do the same. Sometimes you just need to write evaluation first before even I try to develop the applications. So for example, I just need to start with like, "Hey, what is a good response is going to look like? What is a bad response look like?" Be very clear on what it is and what success look like. For example, people call it maybe the number of tickets you can automate, but what success could look like from the beginning before we go in.
But also keep in mind is that is a sort of story of searching for the key there's a lamppost, right? Because we see that somebody lost a key and they search under the lamppost because it's just this light there. So I do think like ROI focus development is very similar sometimes with development applications, not because it maybe the most impactful but just because it's the easiest one for us to measure ROI, and which means that we are just missing out on potentially an application it can be very impactful in data life. We just can't have a clear way to evaluate returns from the beginning.

Denys:
For sure. A lot of great stuff there. We'll definitely agree with both of you. Chip, I definitely agree, enterprises should be willing to take on that little bit of risk there, have some budget to play around and also think about what the most important problem is, even though it's not always the

easiest one to solve. And Colin, 100% agree, need to continuously evaluate things. I think even internally for us, when we're using vendors or thinking about our own product, thinking are we solving problems the best? And I think right now is a great time to shake things up in the industry as sort of these cool new models are coming out, technologies are coming out, always being willing to evaluate new tools and I think if you as a company have a great product and have a great relationship in being responsive and adding features, you'd be happy for your customers to do that.

So when we're working with Colin and his team, it is great to see him constantly pushing and asking those kinds of questions because it forces the product to be better. So I think that's a really great mindset, really great way to build a good user experience.
So we have five minutes left. Just wanted to wrap up on some Q&A. I think we've answered most of the questions here. Jean or Jean asked about the inflection point about using your own model versus third-party API. I think that inflection point is pretty far along. At Voiceflow we also have a similar budget for large language models sort of in the tens of thousands. For that it's still not worth it to run our own models. So it's pretty expensive. It takes up time. So I think definitely more than that. Question about the resolution chart for the Trilogy use case. Let's go back to that. Right here. The horizontal axis is time here. So from week one to 19 and then the Y-axis is that resolution rate.

Colin:
Yeah, and just to clarify, that's resolution 100% by AI. We also have AI assisted or Copilot, but we don't track that in this chart.

Denys:
Yeah, yeah, great addition there. Thanks for the detail, Colin. Other item there? Okay, another question about local hosting versus compute. All right, so we have a few minutes left. I think the last thing for us to mention here for this event is that we will be sending out a spreadsheet to everybody based on this event. The spreadsheet will be based on cost estimations for RAG. One of the topics we talked about. Cost of using different models, getting into those use cases. So we do want to leave you with a small tangible item to talk about. So we're going to send that link. It's going to be a Google sheet. You can copy it, play around with it. If you have any questions, reach out to myself or the Voiceflow team, we can walk you through it, go through a session.
But otherwise, really appreciate everybody joining us here today. Big thank you to the panelists. Thank you Chip and Colin, really appreciated that discussion. Talking about business, talking about tech, diving into the details. It's been a great session and thank you everybody for joining and asking your questions. We're really interested to hear how this topic continues to evolve. So keep in touch and keep asking those good questions about ROI and budgeting. So thank you everybody, and we'll see you on our next panel.

15 KPIs for Measuring and Scaling a Generative AI Strategy

https://www.voiceflow.com/blog/15-kpis-for-measuring-and-scaling-a-generative-ai-strategy

Denys Linkov

Expectations for AI transformation plans are immense, with 58% of CEOs expecting product improvements in the next 12 months. Balancing this short term POC pressure with a 5 year AI strategy is challenging.

There are several paths to a first POC, and the leaders that prioritize a well-built first experience with a clear path to scaling to other use cases will create competitive advantage in this AI automation race.

In this report, we'll share key milestones, frameworks, and KPIs that will help you align teams to deliver AI solutions. This journey starts with an initial bespoke use-case leading to a mature iterative product development organization.

Initial AI journey and Proof of Concept

KPI	Pitfalls	Recommendation
Time to stand up a cross functional team	Over indexing on the AI component	1 product owner 1 UX 1 developer 1 ML engineer
Time to clear problem statement	Skipping the step	Choose a simple first use case: Generative support, internal FAQ, data enrichment
Time to user interviews	Skipping the step	Curate a set of sample users when the project starts
Time to first demo	Waiting for perfection or splitting tooling between business team members and technology teams	Choose tooling that lets the cross-functional teams collaborate on their build with each iteration

Your initial AI project should be about clearing the list of unknowns. It's essential to iterate on ideas and demos to create a first implementation. This work falls into three buckets - team coordination, delivery speed and user iteration speed.

As a leader, your core objective is to unblock teams and provide a space to iterate and improve AI products. Each of these objectives should come with their own key performance indicators to track and measure time until delivery.

Leaders need to structure a team that's set up for success. Many generative AI projects have been staffed exclusively within a ML or IT team, ignoring cross-functional needs to build strong customer experiences and corresponding business cases. This team should be capable of deep diving into a problem area and creating clarity in a fast paced space. Although not a strict list, successful POCs typically have inputs from a product owner, UX, developer and ML engineer.

Clearly defining a problem statement for your POC will help teams deliver with focus and avoid getting lost in AI distractions. In many circles, AI has become a solution looking for a problem, which is great for initial momentum, yet returns in fury and creates lasting tech debt in years 2-3 of a 5 year AI strategy.

User interviews often forgotten in the push to build. Curating a beta group is a highly valuable way to measure product improvements, whether these users are internal or external. These iterations should be quick, with notes shared across the entire team to understand key concerns and positive outcomes. Often times this step is skipped as engineering teams focus on the build and only at the last moment deliver a sharable experience.

The most important item is **getting a cohesive demo prepared**, including a clear value proposition and budget for taking the project to the next step. The initial proof of concept is designed to de-risk a project, but success is measured by projects' progress. Without progress many teams get stuck in proof of concept purgatory, where these early use cases keep getting built with no clear next step for production or past first launch.

Presenting a cohesive narrative for moving the POC to production becomes essential.

Role	Responsibilities
Leadership	1. Create an impactful, cross functional team 2. Choose a business area to automate as a first use case 3. Build an environment of urgency and strong customer focus 4. Have a clear plan and ask to proceed to production
Product	1. Gather requirements 2. Prototype refinement and team facilitation 3. Iterate and user test 4. Prepare final demo 5. Project scoping
Design/UX	1. Design and prototype creation 2. Leading user research
Engineering	1. Create first iteration of the application 2. Basic front end and branding 3. Mimic data sources and data 4. System architecture 5. RAG document collection
Machine Learning	1. Select techniques based on use case 2. Test prompts 3. Test LLM and NLU models 4. RAG design

Deploying to Production

KPIs:

1. *Time to connect to production data*
2. *Time to move between product environments*
3. *Time to build an evaluation suite*
4. *Time to finalize UI*
5. *Time to complete security and risk assessments*
6. *Time to launch feature*

As clearer use cases are defined, more teams become involved in the process. With more teams, clear communication becomes essential for progress and alignment. To navigate this complexity, it’s important that feature and product launches are well managed and the tooling reflects this collaborative, scaled effort.

Connecting to production data is essential at this step. In a proof of concept environment we often focus on mimicking production data or using a subset since integrating with existing systems can be challenging. When moving into production, this should be a key task for the engineering team to connect and test the POC with production data. Depending on the organization, this data may sit in a higher environment, requiring the team to deploy to an user acceptance testing environment (UAT).

Next, teams should focus on **speed to move the features through each stage of the product environment**. Typically there are 2-4 environments depending on the stage of the company, but similar to software projects, fast progress times through environments leads to faster iteration cycles.

On the engineering side of the house, the team should be measured on **creating strong testing metrics and brand aligned front-ends**. Testing is particularly challenging since generative AI models are non-deterministic, requiring tests to validate how users can interact with the models. This work will typically be done in conjunction by a user testing, ML and engineering team to define and deliver on thorough and relevant tests. When considering the front end, generative AI applications should feel like a natural extension of an existing application, rather than being bolted on as an afterthought. This is where design, UX and front end teams need to collaborate to create the integration. To speed up the delivery process, using pre-existing open source or hosted front end frameworks can be strongly beneficial.

KPI	Pitfalls	Recommendation
Time to connect to production data	Waiting until final round of testing to connect to production data sources	Deploy a simple version of the application to a production environment to confirm data and LLM usage
Time to move between product environments	Moving between different environments is only done once.	Move between environments as quickly as needed to parallelize tasks across 8+ teams
	Staging environments (qa, uat) do not mirror production	Create environments that are as similar as possible
	Stuck refactoring code to make production grade	Use libraries and best practices, scope refactoring and production readiness into estimates
Time to build an evaluation suite	Skipping the step, not being rigorous in testing	Define top 20 key user behaviors. Testing should be a combination of user testing and technical LLM evaluation.
Time to finalize UI	Rebuilding from scratch	Use an existing set of components, platforms or libraries
Time to complete security and risk assessments	Delaying conversation until launch	Involve security and risk once the initial problem space has been chosen, make sure CEO and security/risk are aligned about opportunities and risks
Time to launch feature	Getting stuck in POC purgatory	Implement project phases and de-risk with continuous deployments, feature flags and testing environments

Beyond product development often lies a negotiation with the security, compliance, and risk teams. Generative AI security and risk policies are still being defined or iterated on within many organizations, and navigating these policies can slow down projects. Ideally these discussions are started in the POC phase to fully align teams early. We discussed some of these approaches in a prior piece.

Launching the feature live is the final milestone. Once the first application goes into production, there will likely be more projects starting, so the KPIs can apply across new projects across the organization. It’s important that leaders have set a clear product roadmap and teaming structure so that each new use case and iteration is not additional technical debt, rather an additive product lifecycle that becomes more efficient with each new launch.

Role	Responsibilities
Leadership	1. Facilitate conversations across many areas of the company 2. Work with engineering and finance to finalize budget and total cost of ownership 3. Augment team with outside capabilities if required
Product	1. Full customization 2. Reusable systems 3. User testing 4. Copy writing refinement 5. Understanding LLM capabilities and rate limits
Design/UX	1. Finalize designs 2. Full system integrations with design
Engineering	1. Testing and testing automation 2. Integrate with existing systems 3. Provision dev, qa, UAT and prod environments 4. Configure feature flags for release 5. Customize UI
Machine Learning	1. Optimizing performance and prompts within a budget 2. Hallucination analysis 3. Intent optimizations 4. RAG optimization
Operations/ Site Reliability Engineer	1. Support team onboarding 2. Operating playbooks 3. Known issue list 4. Entitlement management
Marketing	1. Project walkthroughs and screenshots 2. Copywriting 3. Working with design to align on brand
Security/Risk	1. Application risk assessment 2. Verify vendor security 3. Application testing

Iterate and Scale

KPIs:

1. *Time to make simple updates*
2. *Time to launch subsequent version*
3. *Time for next team to launch product*

4. *Cost per additional project*
5. *Total cost of ownership*

After deploying at least one use case to production, the challenge for an organization is to scale their generative AI applications across teams in repeatable and financially sustainable way. Competitive advantage is created when teams can build with an innovation mindset and not be stuck in maintenance mode. The KPIs for this phase focus on this agility and cost.

It's first important to measure the **ability to quickly update your AI applications**. Simple changes like minor prompt updates, copywriting, or API version bumps should not be complex processes. Similar to rapid software release, making small changes to generative AI updates, especially after initial release, builds trust with customers and address minor oversights.

What about **larger iterations**? Models, data and product requirements continue to evolve quickly, so being able to update features and capabilities will future proof your generative AI products from becoming legacy product lines (e.g. an AI chatbot on your homepage that was launched as an initial use case, yet quickly becomes ignored and outdated without a proper team, workflow, and iteration cycle).

KPI	Pitfalls	Recommendation
Time to make simple updates	Waiting until major releases to make product updates.	Set up deployment pipelines and processes to allow daily updates. Allows models, data and fixes to be pushed quickly to maintain momentum.
Time to launch subsequent version	Not planning improvements after initial product release.	Thinking about Gen AI as a product rather than just a feature.
Time for next team to launch product	Teams work in parallel not leveraging learning from initial launches	Clear communication and reuse across the organization, with a decision making framework to onboard onto existing versus new solutions.
Cost per additional project	Not realizing a lower marginal cost after an initial use case. Building a centralized platform that does not apply to use cases.	Build or buy a solution that will let you onboard and scale more use cases across a business
Total cost of ownership	Not budgeting organization-wide or looking for reuse savings	Looking at costs in use case depth breadth as they scale and measuring ROI each

Similarly, as adoption increases across an organization through new use cases, teams should be able to **onboard more quickly** after the initial use case. The total cost of ownership for each new team and use case should become more efficient as the level of innovation and automation across an organization ramps exponentially. **At the CIO level, this should be an essential metric when implementing and scaling your AI strategy.**

The final two KPIs in this section focus on cost, both the marginal cost per application and the total cost across a strategy. These two metrics help organizations plan future projects and measure the ROI across their generative AI initiatives. They also help scope investments to be leaner and more deeply integrated within a technology initiative, rather than a separate line item with a large commitment.

Role	Responsibilities
Leadership	1. New use case enablement 2. Onboarding a new line of business 3. Measuring ROI
Product	1. Product launch retrospectives 2. Multi modal iteration 3. New use case roadmap 4. Improved analytics and reporting
Design/UX	1. Designs and prototype creation 2. Leading user research
Engineering	1. New integrations 2. Data refresh 3. Iterative integrations
Machine Learning	1. Adapting to new models 2. Prompt iterations 3. Model version changes
Marketing	1. Copy tweaks 2. Product update copy

Conclusion

As teams build their generative AI strategies and adopt their core workflows, planning each phase of the process is essential. While generative AI feels like an immediate fire to address, projects and capabilities will continue to evolve and require a comprehensive 5 year strategy.

The leaders and teams that execute a well-built first POC that's anchored in a strategy for scaling to other use cases will create competitive advantage in this AI automation race across industries.

Citations

PWC, 2024: https://www.pwc.com/us/en/library/ceo-survey.html

- We have more customers with live agents in production and they want to know how to improve them
    - How can they prove the value of AI agents? Show time/cost savings to leadership?
    - Kelly answers similar questions over and over again—how do you know when to expand to a new use case? Is it working well enough to expand its capabilities?
- LLMs interacting with customer is still new for most customers. Enterprise is getting on board and having to track what parts of their AI agents are useful and when to expand use cases, so they need to know how to measure their agents' success