GO BEYOND CHATBOTS

Designing multimodal systems

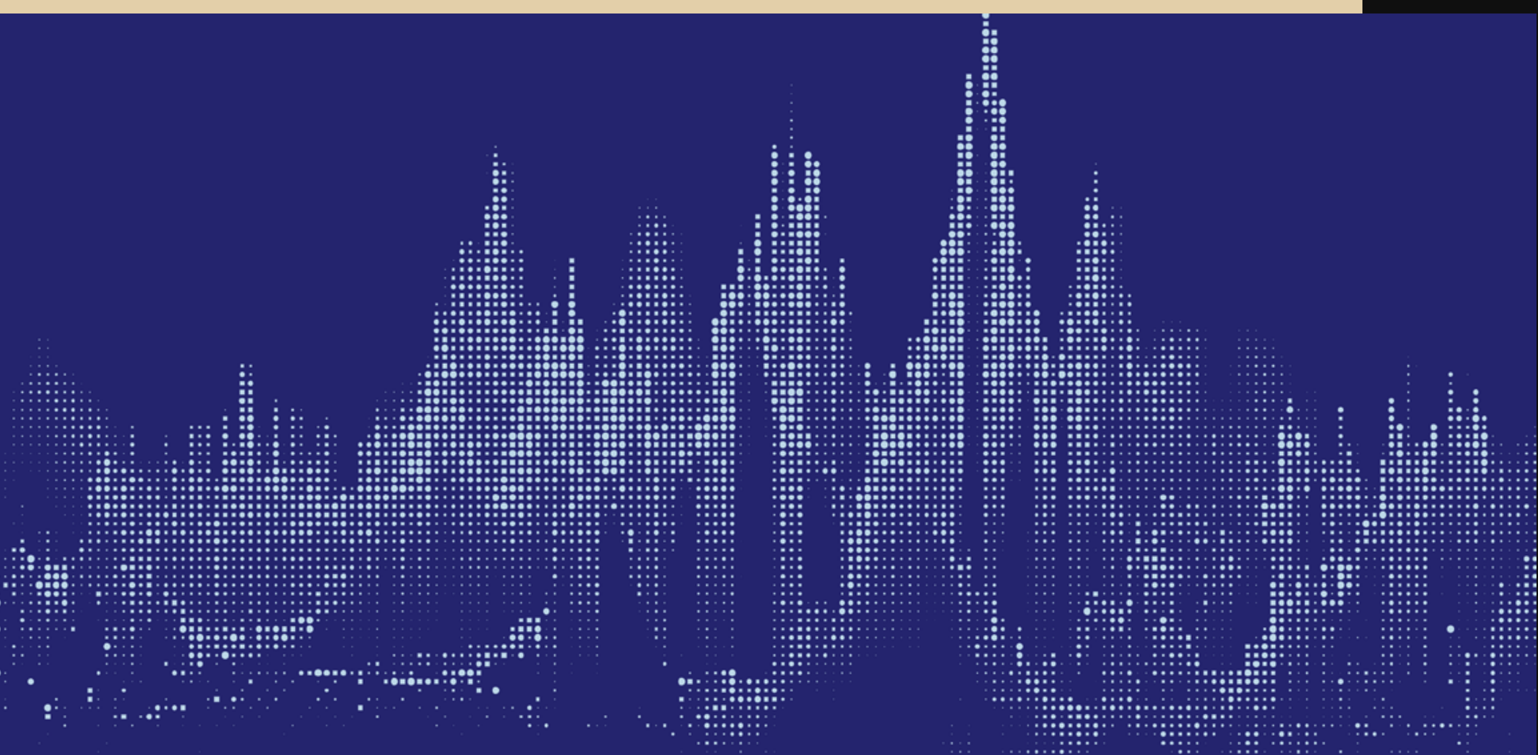


Table of contents

Executive summary

Multimodal systems rarely fail because the model isn't big enough. They fail because labs try to treat 'vision + audio + sensors' like slightly awkward text, and discover too late that they've built the wrong data, the wrong pipeline, and the wrong evaluation regime for the task they actually care about.

It's easy to think of multimodal as a bolt-on to language models, but the causality largely ran the other way. Modern deep learning took off first in vision: large, carefully curated image datasets and benchmarks (most famously ImageNet, led by Fei-Fei Li and collaborators) forced the field to confront scale, annotation quality, and systematic evaluation. The same pattern of big labeled corpora, standard benchmarks, and aggressive iteration later became the template for large language models.

Attention followed a similar path. Before it became standard in text-only transformers, it was proving itself in image captioning and vision-language models that had to decide where to look in an image while generating text. In that sense, many of the tools that now dominate language modeling were stress-tested in multimodal work first; the new challenge is turning them into robust, end-to-end systems that can handle messy real-world data.

Perception-heavy systems like self-driving stacks and industrial robotics were pushing vision, sensor fusion, and control years before large language models emerged, largely independent of NLP. What's changed recently is the coupling: strong language backbones now make it cheap to connect modalities (text↔image, text↔audio, text↔action), so previously separate vision or robotics pipelines can be steered, described, and evaluated through a common linguistic interface.

This paper is written for teams that already know how to ship text-only models and are now being asked to "make it multimodal". We argue that the hard problems are: deciding which real-world tasks are worth modeling; designing datasets and schemas that reflect those tasks (not just benchmarks); choosing how and where modalities should interact in the system; and evaluating models in the messy regimes where they are most likely to break. The architecture is the last step, not the first.

The stakes are more than cosmetic. A bad text model wastes time or erodes trust. A bad multimodal system can misinterpret a road scene, mishandle medical imagery, or learn the wrong rules from the wrong sport and feed systematically wrong decisions into automated workflows. These are failures of design and measurement long before they are failures of optimization.

We lay out the approach we've found useful in practice: start from the task and error costs; inventory and align your data before you chase a benchmark; treat pipelines as engineered systems with explicit fusion and failure points; and use evaluation as a three-legged structure of benchmarks, targeted red teaming, and fine-tuning loops that answer five concrete questions about model choice, safety, readiness, deployment, and monitoring.

The guidance here is distilled from work with foundation model labs, robotics and perception teams, and enterprises trying to deploy multimodal systems under real constraints. The sections that follow unpack this approach in more detail, using concrete examples, common failure modes, and design patterns for building multimodal systems that survive contact with the real world.

01

Designing outcomes for your multimodal model

Why multimodal isn't just "more data"

The processes and disciplines behind successful multimodal machine learning are very different from those used for a single modality. Multimodal models and their infinite combinations of data add another dimension that amplifies the types of problems we see already with text models, like safety, reliability and accuracy. The expansion from two to three dimensions means an explosion in the combinations of data types, vastly increasing potential points of failure or bias. That's why labs can't just "start experimenting". They need to whiteboard the system first: what the model is for, which modalities really matter, how the data actually lines up, and where failures would be unacceptable.

So what changes when you go from text-only to multimodal? Suddenly you're not just modeling sequences, you're modeling synchronized, lossy, and uneven streams of information with different failure modes and annotation costs. Every decision—what to label, how to align modalities in time and space, where to fuse signals, how to evaluate errors—turns into a systems design problem, not just a modeling choice. In practice, your bottlenecks move from "do I have enough tokens?" to "do I have the right signals, aligned at the right granularity, to support the task I actually care about?"

Optimize for tasks, not benchmarks

Many engineers rush towards testing and building models without sufficient time spent in ideation: understanding customer needs, surveying available data, and designing their projects accordingly. Ask the question: is this something a customer would want, find valuable, and be willing to pay for?

Early, thorough whiteboarding and inventory of available data assets is crucial to ensuring that the eventual AI solution addresses genuine business or customer needs. A lot of multimodal projects are designed to score well on benchmarks, rather than to be adopted. There are so many attributes that are easy to measure, like accuracy, or efficiency, but that can leave researchers afraid to test for what really matters. Ask tough questions like: can I get this AI model to summarize a call and identify the issue? Can it look at a meme and tell me why it's funny?

> The true success of a model is: "does it do the job well for the end user?" not "did it get good grades?" We don't tell people our SAT scores 10 years after graduating high school; models shouldn't cite their benchmark scores months and months into doing real work.

There's rarely a clear, definitive moment when your multimodal data is 'perfectly ready'. Progress is iterative. Hypothesize (is my data clean enough?), test, analyze results, and return to data or model adjustments as necessary. This might seem as laborious a task as Sisyphus pushing the rock uphill, but each time, you're trying to make the burden lighter and get closer to the summit.

02

Inventory, align, and curate your multimodal data

"When we're talking about all this imitation learning, it's really wildly inefficient. Like if you gave somebody 10,000 books to learn something and they only learned a little bit of it, you'd be pretty embarrassed... I do think as we get into more experiential learning, as we're getting into the levels of reinforcement learning, we're gonna be able to see a much smaller number of exquisitely performed tasks, training our models instead of such a very, very large amount of data."

Rob Ferguson
CTO, Microsoft for Startups

Beyond schema: Seeing the quirks in your data

In multimodal settings, data comes with a much greater variety and volume. A fundamental step is to know your data: not just its format, but its quirks, imbalances, and hidden assumptions.

A common multimodal failure mode we see is teams believing they're 'data rich' because they have images, logs, audio, and text, but almost none of it is aligned at the sample level. The project trains for months, but the model never really becomes multimodal; it just learns separate unimodal patterns with no way to fuse them. Only after disappointing results do they realize the real blocker wasn't model architecture, but the absence of overlapping, well-synchronized examples across modalities.

You might already have what you need: want to improve your customer experience? Those 100,000 hours of audio recordings and transcripts of customer service calls could be a valuable source.

Annotation: The bottleneck you can't ignore

Data annotation deserves special attention.'Quality over quantity' should be your guiding mantra. It's better to have 200 perfect annotations than 2,000 with noise. Over-labeling (labeling more features than you initially think necessary), will modestly increase up-front costs, but it's far more efficient than retroactively relabeling dataset portions. For example, in a manufacturing-quality project, labeling not just the item but also the machine, conveyor segment, nearby tools, and operator hands can save considerable time and resources further downstream.

02: Inventory, align, and curate your multimodal data

Recent fine-tuning work by Ouyang, J,. et al.[1] shows that even modest fractions of subtly incorrect data (on the order of 10-25%) can significantly degrade domain performance and induce misalignment, with a fairly sharp threshold on how much 'bad' data a model can tolerate in practice. For multimodal regimes—where each labeled example is far more expensive—that tolerance is effectively lower still; you can't rely on scale to wash the noise out.

How you structure labels and IDs matters as much as raw volume:

Aligned multimodal tuples

For 'true' multimodal, ensure samples share a stable ID across modalities (e.g. video frame, audio slice, transcript span, metadata row), so you're not stuck with lots of multimodal data that doesn't actually overlap.

Event- and action-centric sequences

For temporal media, don't just store raw audio/video; explicitly represent events and actions with timestamps and state (e.g. 'door opens', 'arm grasps object'), so you can later train controllable models, not just passive perception.

Egocentric task traces

For robotics / procedural work, capture sequences (sensory stream, human narration, control actions) rather than one-off snapshots; a few well-labeled task executions beat thousands of ambiguous clips.

Don't just 'label the data'; design how expertise flows into the dataset:

Start tiny, with experts

Pilot your schema on a very small slice with true domain experts (radiologists, sports analysts, operators), then freeze the guidelines before you scale to a larger pool.

Separate 'what is' from 'what to do'

Keep perception labels (what's in the frame / audio) distinct from judgment labels (approve loan, escalate call), so you can debug bias and safety issues later instead of baking them into a single opaque tag.

Build a grader layer

Use a second tier of annotators to score or correct demos and labels (Was this tele-operation sequence 'expert'? Is this caption faithful to the video?), so the model learns from your best data, not just your most data.

[1] Ouyang, J., et al., "How Much of Your Data Can Suck? Thresholds for Domain Performance and Emergent Misalignment in LLMs," 2025.

02: Inventory, align, and curate your multimodal data

In multimodal, quality isn't optional

You also need to recognize when your dataset is unbalanced: say, overrepresenting one demographic or object class. Careful weighting, targeted sampling, and continuous statistical monitoring are necessary to avoid encoding bias into your models.

If you're coming from text only, you need to challenge a lot of your instincts. In text-only land, you can often get away with 'just add more data' and cheap augmentation. In multimodal settings, every extra sample is expensive (video, sensors, expert labels), and most of the value comes from how well modalities are aligned and annotated, not how many raw hours you've scraped.

> "The cost of incorrect data is heavy, highlighting the critical need for extremely high-quality data curation or, alternatively, leveraging robust base models without unnecessary fine-tuning for high-stakes applications."

Jian Ouyang
Model Behavior Engineer & LLM Researcher, Invisible Technologies

Concrete multimodal pathologies to watch for include:

- Misalignment across modalities (e.g., audio transcripts that are a few seconds off from the video)
- Label leakage (e.g., captions that literally state the label, so the model never has to 'look' at the image)
- Representation collapse, where one modality (usually vision) dominates the shared latent space and the others barely move the needle.

Multimodal AI relies on large, high-quality, well-labeled datasets. Data curation is more difficult with heterogeneous sources. Labs should:

- Employ human domain experts for labeling complex data types (e.g. medical images, audio clips) for accuracy
- Be strategy agnostic: willing to clean, augment, or redesign their data capture pipeline based on the specifics of each application
- Choose data partners capable of supporting this flexibility, rather than being tied to a single labeling or augmentation technique.

Data infrastructure and pipeline design should be considered foundational. Building robust systems for ingesting, cleaning, and storing multimodal data from the onset prevents much more expensive problems downstream.

02: Inventory, align, and curate your multimodal data

Set the guardrails: Ethical, legal, and safety constraints

Multimodal AI almost always processes personal or biometric data (like voice or images), so during problem definition and data design, labs need to address questions like:

What exactly are we allowed to represent?

Are we storing raw faces, raw voices, or only derived embeddings? Are we stripping EXIF, GPS, or background audio that could re-identify people or locations? These choices determine what your latent space can encode.

Which jurisdictions and regimes apply at each stage?

A single sample can cross GDPR, CCPA, HIPAA, or sector-specific rules as it moves from collection → labeling → training → deployment. Your data flows and retention policies need to be drawn with these boundaries in mind, not patched in after an incident.

Where does consent live in the pipeline?

Is consent tied to the raw media, the derived features, or the downstream tasks? Can you handle a withdrawal of consent through your storage, training, and evaluation stack, or would you have to rebuild everything from scratch?

What can be inferred by fusing modalities?

Combining innocuous signals (room audio + low-res video + metadata) can reconstruct identity, health status, or location in ways no single modality could. Model and data design should assume cross-modal inference.

Thinking about these issues at the start of the project will likely incur extra cost, but it saves money downstream on costly redesigns or even legal jeopardy.

03

From “one big model” to decomposed systems

Multimodal perception is harder than it looks

When starting out on the multimodal journey, setting expectations with all stakeholders is vital: and the first of these is around the difficulty of the task. **Moravec’s Paradox is a helpful guide: tasks that humans find cognitively complex, like solving a tricky math puzzle or winning at chess, are relatively easy for an AI model to accomplish. But an action that humans do unconsciously: spotting a familiar face across a crowded room, or picking up an object from a table, is far less straightforward for a machine or a robot.**

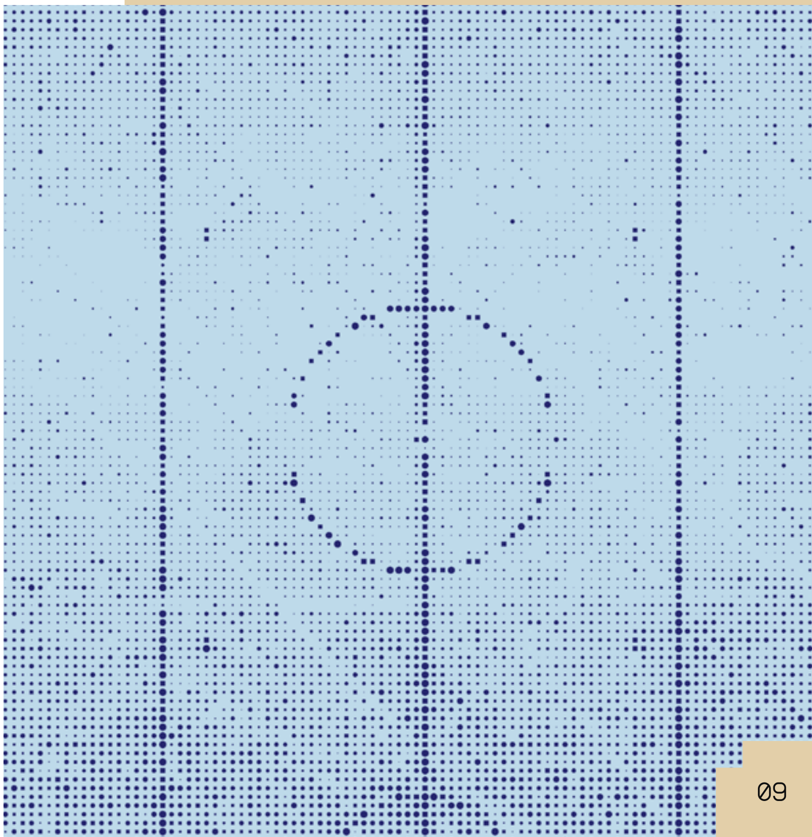
Take a multimodal system trained to analyze player movement. In hockey, offsides depends on the puck crossing the blue line before attacking players; in football there is no equivalent ‘blue-line zone’, and defenders can move across the line of scrimmage in many situations.

The model repeatedly transferred hockey-style priors into football: treating yard markers like blue lines, inferring that ‘defense can’t cross X line’ as a universal rule, and miscalling situations that depend on temporal context (like offsides in hockey, which is sequence-dependent, not frame-based). Under the hood, it was juggling vision (markings, field/rink geometry, uniforms, puck/ball state), text (rulebooks and exceptions), and audio (whistles, commentary), plus tight synchronization so rules applied at the right moment. Fixing this required sport-specific ontologies (players, zones, lines, possession), curriculum training with cross-sport counterexamples to prevent rule bleed-over, temporal models that reason over sequences, and evaluation suites per sport that stress rare edge cases like delayed offsides and motion penalties.

It takes multiple degrees of model tuning to teach it to ignore ‘noise’ like the fans in the bleachers as opposed to the players on the field.

Decompose the task, don’t rely on one world model

This is where task decomposition beats a single end-to-end model. You can either ask one giant network to go from raw broadcast feeds to ‘good offsides and motion calls across all sports’, or you can break the problem into homography, detection, tracking, role/zone assignment, and rule application. In practice, the latter is easier to debug and adapt: when football starts inheriting hockey’s blue-line priors, you know exactly which submodel and ontology layer to fix, instead of trying to interpret a monolithic ‘world model’ that has blended the two games together.



03: From “one big model” to decomposed systems

Beware the “just add vision” reflex

Tying data practice or model choice to a single tool or method sets you up for failure when facing unforeseen data challenges. You’ve heard the saying “if all you have is a hammer, then every problem starts looking like a nail”? When it comes to multimodal AI, you’ll need to approach different problems with a screwdriver, wrench, or drill, as needed. Don’t fall into the trap of assuming a technique that worked for text or images will apply to audio or video. Each data type, and even each segment of your pipeline, may demand distinctly tailored models or algorithms.

The build-out of the end-to-end solution is just as important as choosing the correct tool for a specific problem. Break your challenge into constituent problems: homography, tracking, detection, segmentation, and so on. Design modular components. Pipelines, where different models interact (sometimes sequentially, sometimes in parallel), allow for flexibility and robustness as new data types and challenges emerge.

Teams that only know text models will try to treat everything as “add a vision encoder and fine-tune”, which is often wrong. The instinct is understandable: in the LLM world, you bolt on a retrieval layer, add some instructions, and call it a day. With multimodal systems, that won’t work.

Typical failure modes:

Treating pixels like tokens

You assume images or video frames behave like longer prompts: more context → better performance. In practice, you overwhelm the model with redundant or noisy visual information and never specify what it’s supposed to perceive or ignore.

Ignoring structure and time

Text models trained on simple token sequences don’t prepare you for spatial layout, object permanence, or long temporal dependencies. If you don’t explicitly model geometry and sequence (e.g. tracking, state, events), you get systems that ‘see’ a lot and understand very little.

Forgetting the rest of the system

In text land, it’s tempting to push everything into the model. In multimodal land, perception, fusion, decision, and control often want different components and different failure budgets. Trying to pack all of that into one fine-tuned backbone gives you beautiful demos and fragile products.

Start from the task and constraints, then decide which parts want a multimodal backbone, which parts want small, specialized models, and which parts don’t need ML at all. The architecture should fall out of the problem, not out of whatever your text stack already knows how to do.

03: From “one big model” to decomposed systems

(Don’t) bring the noise

Noise is the ever-present adversary in machine learning, and it comes in various forms:

- **Input noise** – ‘total noise’ like empty frames, motion blur, or sensor glitches that are irrelevant to the task.
- **Label noise** – ‘partial noise’ like loose bounding boxes, inconsistent class labels, or annotators disagreeing on events.
- **Alignment noise** – modalities that are out of sync (audio a few seconds off video, transcripts misaligned to frames, mismatched IDs across streams).

Understanding and minimizing noise through expert annotation, feature engineering, and intelligent model training is critical.

Some degree of noise is inevitable, especially with video, audio, and sensors, so the goal is robustness rather than purity: architectures that can ignore empty or corrupted regions, loss functions and sampling that down-weight dubious labels, and augmentation that reflects the real degradations your system will see in the wild.

03: From “one big model” to decomposed systems

Factor in bias and safety in practice

Bias in multimodal systems isn’t just ‘underrepresented faces in the dataset’. It shows up in which modalities you have, how they’re captured, and who labeled them:

- Cameras tuned for some skin tones more than others.
- Microphones and ASR that degrade on certain accents or noisy environments.
- Text and audio drawn from a narrow socioeconomic or geographic band.
- Labelers who bring their own priors into “what’s happening in this clip” or “what should happen next”.

Balance your training data via sub-sampling, up-sampling, and weighting strategies. Critically, ‘understand your data’ again comes to the fore: you cannot correct for what you haven’t measured or are unwilling to see. Compare error rates, calibration, and abstention behavior across demographics, geographies, environments, and device types. In multimodal work, that often means conditioning on both who is in the scene and how the scene was captured (camera, mic, channel).

If your system is used in higher-stakes settings like finance, medical, navigation, or security, you should know where it is systematically worse, and decide in advance what happens there: abstain, fall back to a simpler model, or route to a human. Treat those choices as part of the system design.

> “When we look at audio models, one of the ways we measure bias is not just in what the user is saying to the model, but what other things exist in that file. If there are sirens in the background of a user input, if there is a certain phonetic annotation, if there’s a lot of children or dogs crying in the background, is that model gonna be less likely to do one action—approve a loan, do a response time… because of the socioeconomic connotations associated with those background noises?”

Lydia Andresen
Executive Director,
Invisible Technologies

04

Data pipelines as engineered systems

Make the end-to-end flow explicit

Successful multimodal systems are the path from raw inputs to decisions. Sketch that path. If you're monitoring animal welfare on a farm, for example, what actually happens between "barn cameras and microphones" and "alert a vet about a distressed animal"? What comes in (video of stalls, audio of vocalizations, temperature sensors)? What happens first, second, third? Which steps clearly depend on each other (e.g., locate animals before assessing posture or injury), and which can run side by side (e.g., processing audio and video in parallel)?

Pipelines must be designed for scale, clarity, and maintainability. Documenting how data flows through your system, from ingestion to prediction and evaluation, makes it easier to iterate as the problem evolves.

Model evaluation in multimodal settings is multidimensional. You'll need to interpret a mix of task-level metrics and modality-specific metrics, rather than a single headline score. High-level system performance should be mapped to concrete business or research goals: precision, recall, specificity, etc.

Critically, use metrics diagnostically. If a model confidently mislabels objects (for instance, it treats fences as humans), it's likely symptomatic of mislabelled data or model architecture problems. Iterative testing, visualization, and root-cause analysis of errors are key to improvement.

> "When you want to get to a product that can scale across the whole population of the world, that tail end of data, that last 5% of knowing how a human behaves with your product or what these kind of edge cases do… the amount of diversity that you need in the population ends up being so much greater and the amount of testing you need to really deliver that experience beyond a demo."

Bryce Schmidtchen
AR/VR Engineer,
ex-Vision Pro, Apple

04: Data pipelines as engineered systems

From single scores to slice-aware evaluation

In multimodal systems, a single headline number is almost always lying to you. You care less about 'overall mAP' or 'top-1 accuracy' and more about how performance shifts across slices: low light vs daylight, crowd noise vs clean audio, camera A vs camera B, underrepresented demographics, different motion regimes. The right pattern is:

- Define the slices that actually matter for your task (environments, devices, user groups, edge conditions).
- Track core metrics per slice (error rates, calibration, abstention) and compare them, not just the aggregate
- Treat big gaps between slices as bugs—they're often where safety, bias, and product risk live.

This is especially true in multimodal evaluation: the failure mode is rarely "the model is bad everywhere"; it's "the model is fine in lab conditions and terrible in three very specific, very important regimes".

Watch out for 'death by sample size', which happens when labs can't afford to run enough evaluations for valid statistical conclusions. Smart methodologies, like getting the most insight from the least data, help make evaluation feasible.

05

Technical challenges: Latent spaces, evolution, and compute

Designing latent spaces for multimodal work

The hard part of multimodal systems isn't "adding more encoders"; it's designing architectures and latent spaces so the whole thing actually works in the real world. You need representations that let you link a photo, a PDF page, and a CRM record; add a new modality later without retraining everything from scratch; and support operations like retrieval, conditioning, and control over time.

Forward-looking latent space design ensures future compatibility and capabilities as requirements evolve.

Compute and memory constraints are real. Processing multimodal data is far more resource-intensive than text alone. Labs should plan capacity not just for initial experiments, but for continual improvement and deployment, understanding that model efficiency will improve over time, similar to advances in automotive fuel efficiency.

Under the hood, you have three broad options for representation:

- **Shared latent space.** All modalities are projected into a common space (e.g. CLIP-style). This is attractive for retrieval and zero-shot transfer, and it forces the model to learn cross-modal structure. The trade-off: if the space is too "one-size-fits-all", some modalities (often vision) dominate and others become second-class citizens.
- **Modality-specific spaces with late fusion.** Each modality has its own encoder and representation, and you only fuse them in downstream heads or decision layers. This is easier to tune per modality and often better for task performance, but you give up some of the elegance and generality of a single shared space.
- **Mixture-of-experts and hybrid schemes.** Different experts (per modality, per task, or per regime) share some parameters but keep specialized capacity where it matters. This can give you the best of both worlds—good zero-shot behavior and strong task performance—but it's harder to train.

Which you choose affects everything else: how you collect and align data, where you plug in new tasks, and how painful it is to debug cross-modal failures.

05: Technical challenges: Latent spaces, evolution, and compute

Designing for evolution, not a one-off launch

Multimodal systems don't stand still. You will add new modalities, new data sources, and new tasks over time. That means treating latent spaces and IDs as long-lived infrastructure, not throwaway experiment scaffolding. In practice, that looks like::

- Use stable, modality-agnostic IDs to tie together all views of the same underlying entity (frame, audio slice, transcript span, metadata row), so you can add another modality later without reindexing the world.
- Keep encoders and heads decoupled enough that you can upgrade a vision encoder, or introduce a new sensor stream, without retraining every downstream model from scratch.
- Version representations explicitly: know which latent-space version produced which evaluation runs and production decisions.

06

Evaluation and benchmarking: Beyond leaderboards

Metrics as diagnostics, not just scores

Robust evaluation is critical and must go far beyond conventional benchmarks. A best-in-class evaluation and improvement strategy for multimodal AI should have three pillars that form a ‘virtuous triangle’:

1. **Benchmarks** should mix standard datasets with scenario-specific suites. If you’re doing sports analytics, that means curated clips for things like occlusions, unusual camera angles, rare rule situations; for medical imaging, it means deliberately including low-quality scans, edge cases, and device variation.

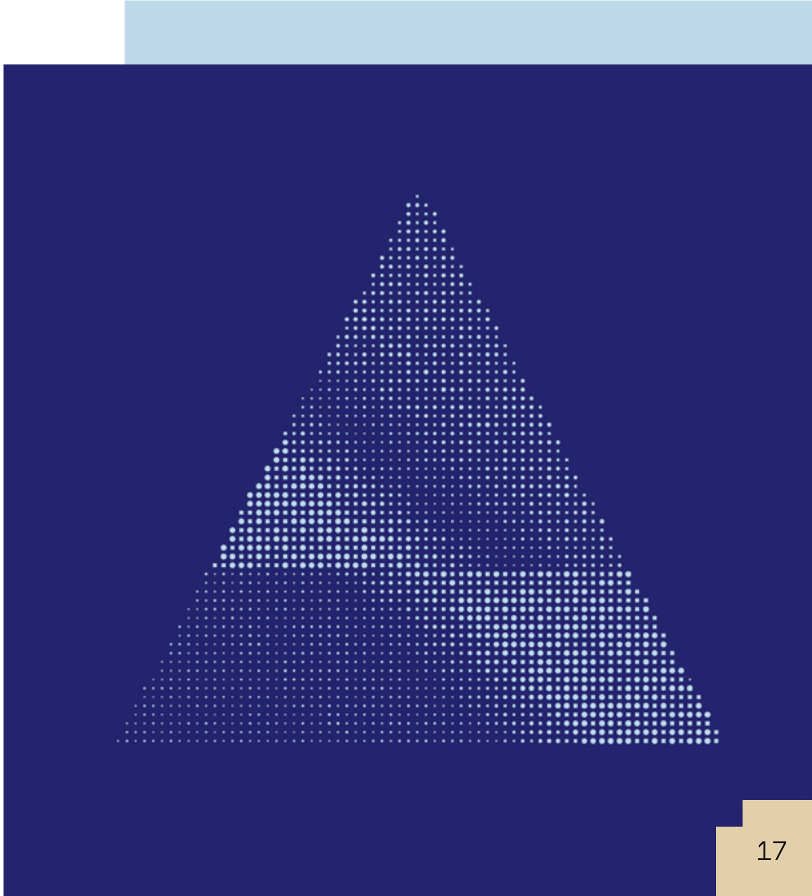
2. **Red Teaming** should explicitly target cross-modal failure modes: misaligned audio and video, conflicting signals between text and image, missing modalities at inference time, or “looked but didn’t see” cases where the object is present in pixels but ignored in the output.

3. **Fine-Tuning** should be driven by what you learn from those diagnostics, not by vibes. When a model repeatedly confuses fences for humans or transfers the wrong rules from one sport to another, that’s a signal to adjust data, labels, or architecture in a targeted way.

A good multimodal evaluation loop always closes the gap between lab benchmarks and field reality. If your triangle can’t explain why a model behaves the way it does on real footage, real microphones, and real users, you don’t have an evaluation strategy yet, you have a leaderboard.

Compute and memory: back-of-the-envelope reality check

Moving from text-only to video+audio is not a 10% bump—it’s often orders of magnitude more data per sample. That gap should directly change how you batch, cache, and schedule training and evaluation. You can’t just reuse your text-era habits: you need tighter control over sequence length, more aggressive downsampling and windowing, and a clear policy for how much multimodal context you can afford to keep ‘live’ at once for a given latency and budget target.



07

Putting it together: A practical pathway into multimodal

For labs and teams that already know how to ship text models, the main risk in multimodal is not “doing nothing”; it’s doing the wrong “something” at scale. A sensible starting point:

01 **Start with the task and error budget, not the modality.**

Write down the concrete decision or behavior you care about, who it affects, and what kinds of mistakes are unacceptable. Only then decide which modalities are genuinely necessary, and what they must add beyond text.

02 **Inventory and align the data you actually have.**

Before you design an architecture, enumerate your real assets: how many samples per modality, how they’re captured, how (or whether) they align at the entity / frame / event level, and where the gaps and biases are. Treat ‘overlapping, well-synchronized examples’ as a separate resource from ‘lots of files’.

03 **Design schemas & annotation strategies to match the task.**

For each task, make explicit choices about label granularity (frame vs event vs sequence), cross-modal links (IDs, timestamps, entities), and who is qualified to label what. Pilot with experts on a small slice, then scale. Don’t assume a generic schema will survive contact with a specific domain.

04 **Choose architecture based on decomposition.**

Decide where you need a multimodal backbone, where specialized unimodal models are better, and where deterministic logic is sufficient. Use end-to-end models when correlation discovery and zero-shot behavior matter; use modular pipelines when debuggability, latency, or safety dominate.

07: Putting it together: A practical pathway into multimodal

05 **Treat pipelines as engineered systems.**

Draw the end-to-end flow: ingest → preprocessing → perception → fusion → decision → output → logging. Specify where modalities are fused, where you can drop or degrade a modality, and where you surface uncertainty or route to humans.

06 **Build evaluation as a recurring process, not a one-off benchmark.**

Combine standard benchmarks with application-specific suites, adversarial/red-team tests, and fine-tuning loops that feed directly back into data and model design. Track performance across slices (environment, device, demographic, modality quality), not just in aggregate.

07 **Make compute and memory explicit design parameters.**

Estimate per-sample footprint for your chosen modalities and context windows, then plan batching, caching, and model size accordingly. If the numbers don't work, reduce the problem (shorter windows, fewer modalities, more modularity) before you start training.

08 **Keep humans in the loop where it matters most.**

For high-stakes uses, define in advance where the system must abstain, where a human must approve, and what feedback you'll collect to improve future versions. Use real user interactions and failures as a primary signal for iteration, not just as anecdotes.

Invisible

A model for making multimodal AI work

Building successful multimodal AI systems is about strategic planning, rigorous evaluation, ethical fortitude, and continual adaptation. **Labs that take the time to assess needs, plan for data and ethics, and invest in robust measurement and improvement practices, will be best positioned to both innovate and deliver solutions that are safe, effective, and widely adopted in the real world.**

If you're ready to move your projects from text to multiple modalities and you need guidance on the right path, contact us today.

TALK TO US