MY WEIRD PROMPTS

Podcast Transcript

EPISODE #54

Tokenizing Everything: How Omnimodal AI Handles Any Input

Published December 11, 2025 • Runtime: 32:58

https://myweirdprompts.com/episode/tokenizing-everything-how-omnimodal-ai-handles-any-input/

EPISODE SYNOPSIS

How do AI models process images, audio, video, and text all at once? Herman and Corn dive deep into the technical complexity of multimodal tokenization, exploring how modern omnimodal models compress vastly different data types into a unified format that a single neural network can understand. From vision encoders to spectrograms to temporal compression, discover the engineering behind the AI systems that can accept anything and output anything.

DANIEL'S PROMPT

Daniel

Hello there, Herman and Coen. So, I was recently taking stock of the different types of multimodal model, multimodal AI model that is capable of processing audio as a modality. And I've doing a lot of work and research into the idea of multimodal AI and a particular use case and trying to sort of evaluate whether audio multimodal models could be a better fit for a task than the more traditional ASR models for speech recognition. And in the course of looking through these models, I came across some omnimodels, and I said, "That's really interesting." There is a task classification on Hugging Face called Any-to-Any. And just as the name suggests, these are models which support any input format and any output format. So, rather than having a model that can take audio, it can process audio as well as text, this can do the full gamut. You can send it a simple text prompt, you can add an image, so it's kind of a vision model. You can add an audio file, and you can mix and match, which creates, each of these really creates a whole world of workflows, but when you sort of stretch it out to Omni, you really get an awful lot of things, and of course the output format in turn also enables a lot of workflow. So, Omni's kind of the far end of it can do anything. And when I was looking at the implementation papers on Hugging Face, which is kind of where there's a line between where geeks at my level of geekery have to sadly drop off and the real hardcore, you know, ML PhDs sort of do their thing. And I was trying to understand that level of it where they were explaining the actual tokenization process by which these different modalities are handled. And that's what I'd love to get into in today's podcast. When we talk about throwing different formats at a large language model, or an AI model, we talked before about text tokenization, the process by which texts are converted into vectors and sent to a model for inference and how statistical prediction happens and then we get, then we get output, which I guess is also vectors getting turned into text so that we can read it. And what I'd be really interested to learn about is how that works for something like video or images, which, just thinking out loud, it seems like we're used to thinking about images as vision models, vision language models, is kind of pretty basic, right? It's been a while since ChatGPT supports that, I think it maybe it even always did. But when I, even if I send a photograph of a can of beer against a white background, that might seem like a really basic photo, but there's a lot of data in that image, more so than text. And when we talk about audio, we're talking about a lot of data representing speech. And when we talk about video, we're talking really, really at the kind of far end of complication where we're sending a large amount of data with 24, potentially, different images every second and asking it to understand based on that. And of course, then we have real-time video. But let's, let's just talk about non-real-time for the moment. That process by which these non-textual data forms go through the same model, how does that work exactly? And how much more complicated is it to for a model to handle these? And it seems to me remarkable, these multimodal models maybe are the kind of forerunners of where we're going to go. But how are some this emerging breed of models able to tokenize all these forms of data?

TRANSCRIPT

Corn

Welcome back to My Weird Prompts, the podcast where we take the fascinating ideas sent in by our producer Daniel Rosehill and really dig into them. I'm Corn, your co-host, and as always I'm here with Herman Poppleberry.

Herman

Hey everyone, great to be here. And I gotta say, today's prompt is exactly the kind of thing that gets me excited because it sits right at this intersection of practical AI work and some genuinely deep technical complexity.

Corn

So Daniel sent us this really meaty question about multimodal AI models—specifically these omnimodal models that can handle any input and any output format. And he's curious about how tokenization actually works when you're throwing all these different data types at a model. Images, audio, video, text, all mixed together.

Herman

Right, and here's what I think is important to understand from the start: most people think of AI models as just text machines, right? You type something in, text comes out. But what's happening now is we're moving toward these systems that can genuinely handle anything.

Corn

Which is kind of wild when you think about it. Like, I can understand how text becomes numbers—we've talked about that before. But when you're talking about an image or a video, that's... a lot more information to encode, isn't it?

Herman

Exponentially more, actually. And that's where the interesting technical problem emerges. So let me set the stage here. Traditional tokenization for text is relatively straightforward. You take words or subwords, you assign them numerical values, and the model processes those sequences. But when you start adding images or audio...

Corn

Wait, but before we go too deep—I want to make sure I'm understanding the scope here. When we talk about these omnimodal models, are we talking about models that are brand new, or are we talking about an evolution of things like GPT-4 Vision that already existed?

Herman

Good question. So there's definitely been a progression. GPT-4 Vision and similar vision-language models have been around for a while—you're right that ChatGPT has supported image input for quite some time. But what's changed recently is the explosion of truly any-to-any models. Models like Qwen3-Omni and Phi-4 that came out more recently can handle audio, video, images, text all at once, and they can output any of those formats too.

Corn

So it's not just about accepting multiple inputs anymore—it's about accepting any combination and producing any output?

Herman

Exactly. That's the real shift. And the reason that matters is because it opens up entirely new workflows. You're not locked into "image in, text out" or "audio in, text out." You could theoretically do audio in, video out with text annotations. You could mix modalities in the input itself.

Corn

Okay, so that's the business value proposition, I get that. But the technical question Daniel was really interested in is: how do you actually make that work? How do you take something as different as an image and audio and convert them into a format that the same model can process?

Herman

Right, so this is where tokenization becomes the key. And I think the best way to understand this is to start with what we already know works for text, and then understand how it scales to other modalities.

Corn

So text tokenization—we've covered this before, but for people who didn't hear that episode, can you give the quick version?

Herman

Sure. A language model doesn't actually understand words the way you and I do. It breaks text down into tokens—could be whole words, could be subwords, could be individual characters depending on the tokenizer. Each token gets assigned a number, and that number maps to a vector in high-dimensional space. So the word "hello" might become token 15342, which then becomes a vector of, say, 768 or 2048 numerical values depending on the model size.

Corn

And those vectors encode meaning somehow?

Herman

They encode statistical relationships. The model learns that certain tokens tend to appear near certain other tokens, and those patterns get baked into the vector space. So "king" minus "man" plus "woman" gets you close to "queen" in that space. It's not that the model "knows" what a king is—it's that it's learned the statistical patterns.

Corn

Okay, got it. So now when we move to images, we can't just do that same process, right? Because an image isn't a sequence of discrete tokens the way text is.

Herman

Exactly. An image is a continuous grid of pixels, each with color values. And here's the problem: if you tried to tokenize an image the same way you tokenize text, treating each pixel as a token, you'd have millions of tokens just for a single photograph. That's computationally infeasible for current models.

Corn

Oh wow, so that's why you can't just throw raw pixel data at these models?

Herman

Right. So what researchers do instead is they use what's called a vision encoder or an image encoder. This is typically a convolutional neural network or a vision transformer—something like CLIP or similar architectures that were trained specifically to understand images. That encoder compresses the image down into a smaller set of tokens that capture the important visual information.

Corn

So you're preprocessing the image before it even gets to the main model?

Herman

Yes, exactly. The image encoder does a kind of semantic compression. Instead of having millions of pixel-level tokens, you might end up with, say, 100 or 200 visual tokens that represent high-level features—shapes, colors, objects, spatial relationships, all that stuff.

Corn

Hmm, but I'm wondering—doesn't that lose information? Like, if I have a really detailed image, you're compressing it down significantly. How does the model maintain understanding of fine details?

Herman

That's a fair pushback, and honestly, it's one of the limitations we're still working with. Different models handle this differently. Some use hierarchical encoding where you maintain multiple levels of detail. Others use adaptive token allocation—the model learns to use more tokens for complex regions and fewer for simple ones. And some newer approaches, like what you see in models that support higher resolution images, use tiling or patching strategies where they break the image into smaller chunks and encode each chunk separately.

Corn

So there's definitely a trade-off happening there between computational efficiency and detail preservation.

Herman

Absolutely. And that's where design choices really matter. A model optimized for speed might compress more aggressively. A model optimized for accuracy might preserve more tokens, which costs more computationally but gives better results.

Corn

Okay, so we've got images figured out conceptually. Now let's talk about audio, because Daniel mentioned that seems like another whole level of complexity.

Herman

Audio is actually fascinating because it's different from images in an important way. Audio is inherently temporal—it's a sequence, like text. But it's a continuous signal, not discrete tokens. So you can't just break it into chunks the way you'd break text into words.

Corn

Right, so how do you handle that?

Herman

Well, typically audio gets converted into a spectrogram first. That's essentially a visual representation of audio—it shows frequency content over time. You can think of it as kind of like a sheet music representation where the vertical axis is frequency and the horizontal axis is time, and the brightness shows intensity.

Corn

Okay, so you're converting audio into something more image-like?

Herman

Exactly. Once it's a spectrogram, you can apply similar techniques to what you do with images. But there's another approach too—some models use raw audio waveforms and apply special audio encoders trained on speech or music. Models like Whisper, for example, can process raw audio directly and encode it into tokens.

Corn

So there are multiple ways to handle audio depending on what you're trying to do?

Herman

Right. If you're doing speech-to-text, you might want to use an ASR-optimized encoder. If you're doing general audio understanding, you might use something more general-purpose. The point is, you're converting that continuous audio signal into a manageable number of discrete tokens.

Corn

Let me make sure I understand the data volume here. Daniel mentioned that a photograph might seem simple but contains a lot of data. Can you put that in perspective?

Herman

Sure. A single photograph might be, say, 3000 by 2000 pixels. That's six million pixels. Each pixel has three color channels—red, green, blue. So that's 18 million individual numerical values. When you compress that through a vision encoder down to, say, 576 tokens, you're reducing it by a factor of about 30,000 to 1.

Corn

Wow, that's aggressive compression.

Herman

It is, but the magic is that the encoder learns to preserve the important semantic information. You lose pixel-level detail, but you keep object identity, spatial relationships, colors, composition—all the stuff that matters for understanding the image.

Corn

And audio?

Herman

Audio is actually even denser in some ways. A minute of audio at CD quality is about 10 million numerical values. When you convert that to tokens, you might end up with a few thousand tokens depending on the model.

Corn

Okay, so now video. Daniel mentioned this as the far end of complexity—24 frames per second, potentially, all encoding different information.

Herman

Yeah, video is where things get really intense. Let's say you have a 10-second video at 24 frames per second. That's 240 frames. If each frame is a 3000 by 2000 image, you're looking at millions of frames worth of data. That's computationally massive.

Corn

So how do models actually handle that?

Herman

There are a few strategies. One is temporal compression—you don't encode every single frame. You might encode every 4th or 8th frame, or you use keyframe detection to identify which frames are actually important. Another approach is to use a video encoder that's specifically trained to understand temporal information, so it can compress not just spatial information but also temporal patterns.

Corn

Like, it understands that if a ball is rolling, the next frame will have it slightly displaced?

Herman

Exactly. The encoder learns temporal patterns—motion, object tracking, scene changes, all of that. So it can encode that information more efficiently than if it was just treating each frame independently.

Corn

But here's what I'm wondering—and I think Daniel was getting at this too—when you have all these different modalities coming in, how does the model actually know how to process them together? Like, you've got these different encoders for different input types. They all spit out tokens. But then what?

Herman

Right, so this is where the architecture gets really interesting. Most of these multimodal models use what's called a unified embedding space. All the different encoders—the text tokenizer, the image encoder, the audio encoder—they all output tokens that exist in the same dimensional space. So a visual token and an audio token and a text token can all be processed by the same transformer layers.

Corn

So they're all speaking the same language at that point?

Herman

Basically, yes. It's like they're all translated into a common currency. The embedding space is the currency. A visual token might be a 2048-dimensional vector, and a text token is also a 2048-dimensional vector. They're different tokens representing different information, but they can coexist in the same model.

Corn

That seems... kind of magical, honestly. How do you even train a model to do that?

Herman

Well, it's not magic—it's a lot of data and careful engineering. The training process typically involves multimodal datasets where you have examples of text paired with images, images with audio, audio with text, and so on. The model learns the statistical relationships between these modalities.

Corn

But wait, that's where I want to push back a little bit. Daniel asked specifically about these any-to-any models, these omnimodal models. When you're supporting truly any input and any output, you're not just talking about having encoders for different inputs, right? You also need decoders for different outputs?

Herman

Yes, and this is where it gets more complex. You need not just encoders but also decoders—or more accurately, output heads—for each modality you want to generate. So if you want to output audio, you need an audio decoder. If you want to output video, you need a video decoder. These are often based on diffusion models or other generative architectures.

Corn

So the core model is kind of this central hub that takes in any modality and outputs to any modality, but it's surrounded by all these specialized encoders and decoders?

Herman

That's a reasonable way to think about it, though some newer approaches are trying to move away from that architecture. Some of the really cutting-edge omnimodal models like Qwen3-Omni are trying to have more unified tokenization schemes so that you don't need as many specialized modules.

Corn

Okay, so let me ask you this: how does that unified tokenization actually work? Because that seems like the hard problem. Text is discrete, images are continuous spatial data, audio is continuous temporal data. How do you find a common representation?

Herman

This is genuinely one of the hardest problems in AI right now. Different approaches are emerging, and I don't think we have a perfect solution yet. One approach is to use vector quantization. You take any input—image, audio, whatever—and you convert it to a continuous representation, then you quantize it into discrete codes. So an image becomes a sequence of codes, audio becomes a sequence of codes, and the model processes codes universally.

Corn

So you're forcing everything into a discrete code-based representation?

Herman

Right. The idea is that if you can represent all modalities as sequences of discrete codes, then the problem reduces to sequence-to-sequence modeling, which transformers are really good at. You've essentially converted the multimodal problem into a translation problem.

Corn

Hmm, but doesn't that lose information? Like, if I'm quantizing audio into codes, I'm losing the continuous nature of the signal, right?

Herman

You are losing some information, but the question is whether you're losing information that matters. If you have enough codes and they're chosen intelligently, you can preserve the perceptually important information. It's a similar trade-off to what we discussed with images.

Corn

Let's take a quick break from our sponsors. Larry: Tired of your tokens being disorganized? Introducing TokenFlow Pro—the revolutionary neural organization system that uses quantum-aligned code clustering to optimize your modality compression ratios. Whether you're processing images, audio, or video, TokenFlow Pro analyzes your data streams in real-time and automatically redistributes your computational load across 47 different dimensional planes. Users report feeling "more efficient" and experiencing what they describe as "improved tensor alignment." TokenFlow Pro also comes with our exclusive MindMeld Accelerator, which you absolutely do not need to understand to benefit from. Side effects may include unexplained confidence in your AI infrastructure and an urge to tell people about dimensional planes at parties. TokenFlow Pro—because your tokens deserve better. BUY NOW!

Corn

...Thanks, Larry. Anyway, where were we? Right, so we're talking about this quantization approach. But I'm curious—when you're dealing with real-time video or real-time audio, does this approach still work? Because Daniel mentioned real-time as being even more complicated.

Herman

Real-time is a completely different beast, yeah. The latency requirements are much tighter. You can't do heavy compression or complex encoding if you need to process and respond to video or audio in milliseconds. So real-time multimodal models typically use lighter-weight encoders and they might process data in a streaming fashion rather than waiting for the entire input.

Corn

So they're making different trade-offs?

Herman

Absolutely. For non-real-time work, you can afford to be thorough. You can do complex encoding, preserve lots of detail. For real-time applications, you have to be fast, which means you sacrifice some depth of understanding for speed.

Corn

That makes sense. But here's what I'm still not totally clear on—when you have this unified embedding space where all modalities are represented as tokens, how does the model actually learn the semantic relationships between them? Like, how does it learn that the word "dog" and the image of a dog and the sound of barking are all related?

Herman

That comes from the training data. If you train on datasets where you have images of dogs paired with the text "dog" and audio of barking, the model learns those associations. The embedding space naturally develops so that semantically related tokens end up close to each other.

Corn

So it's learning through exposure to paired examples?

Herman

Yes. And this is why having good, diverse, multimodal training data is so important. If your training data is biased or limited, the model's understanding of cross-modal relationships will be too.

Corn

I want to push back on something you said earlier though. You mentioned that some of the newer models are trying to move toward unified tokenization schemes to reduce the number of specialized modules. But I'm skeptical that you can actually do that effectively. Don't different modalities have fundamentally different characteristics that require different handling?

Herman

That's a fair point, and I think the reality is more nuanced than I made it sound. You're right that images and audio have different properties. But the idea isn't that you use identical processing for everything. It's that you use a unified framework that can be adapted for different modalities. Like, a transformer can process text, images, audio, video—the core architecture is the same, but you might adjust the input encoding and the output decoding.

Corn

Okay, so it's more about unified architecture rather than unified processing?

Herman

Right. The transformer's strength is that it's agnostic to the type of sequence it's processing. Give it a sequence of tokens, it'll process it. Those tokens could represent text, images, audio, whatever.

Corn

That actually makes more sense. So the real innovation is finding ways to convert different modalities into token sequences that the transformer can handle?

Herman

Exactly. And different teams are experimenting with different approaches. Some use vision transformers for images, audio transformers for audio, and they all output to a common token space. Others are trying to develop more unified encoding schemes. It's an active area of research.

Corn

Let me ask you a practical question. If I'm a developer and I want to use one of these omnimodal models, what am I actually dealing with in terms of implementation complexity? Is this something that's accessible to people who aren't deep ML researchers?

Herman

Well, it depends on what you're trying to do. If you just want to use an existing model like GPT-4o or Qwen3-Omni through an API, it's pretty straightforward. You send in your multimodal input, you get back your output. The complexity is abstracted away.

Corn

But if you want to fine-tune one or understand how it works under the hood?

Herman

Then you're getting into the weeds pretty quickly. You need to understand how the different encoders work, how to handle the token sequences, how to manage the computational load. It's definitely PhD-level stuff if you want to do it from scratch.

Corn

Which is kind of what Daniel was pointing out—there's this line where the documentation on Hugging Face goes from "geek-accessible" to "real hardcore ML researcher territory."

Herman

Yeah, and I think that's actually a sign of healthy development in the field. It means there are tools at different levels of abstraction. You can use these models without understanding the deep internals, but if you want to, the research is available.

Corn

Alright, we've got a caller on the line. Go ahead, you're on the air. Jim: Yeah, this is Jim from Ohio. So I've been listening to you two go on about all this tokenization business, and I gotta say, you're overcomplicating it. Back when I was working in IT, we didn't need all these fancy encoders and decoders and embedding spaces. You just moved data around.

Herman

Well, I appreciate the perspective, Jim, but I think the difference is that back then you were moving data around without trying to extract meaning from it. These models are trying to understand the content. Jim: Yeah, but that's exactly what I'm saying—you're making it too complicated. My neighbor Dennis has a camera system that processes video just fine, and he didn't need any of this quantum code clustering stuff or whatever. Also, it's supposed to snow here in Ohio next week, which is ridiculous for October. But anyway, the point is, can't you just convert everything to the same format and be done with it?

Corn

I mean, that's kind of what we were saying, Jim. You do need to convert things to a common format—tokens. But the trick is that you can't just convert an image to text or vice versa without losing information. You need these intermediate representations. Jim: That doesn't sound right to me. In my experience, computers are pretty good at converting between formats. Why is this different?

Herman

Because you're not just converting format, you're preserving meaning. You could convert an image to text by just listing pixel values, but that doesn't preserve the semantic meaning—the fact that you're looking at a dog, for instance. These encoders preserve that semantic information while reducing the data volume. Jim: Hmm, I guess I can see that. Still seems unnecessarily complicated to me though. Anyway, thanks for explaining, I guess. My cat Whiskers is staring at me weird right now, which is concerning. But yeah, I'm not convinced you need all this complexity, but I'll take your word for it.

Corn

Thanks for calling in, Jim. Jim: Yeah, no problem. Keep up the podcast or whatever.

Herman

So one thing that's worth thinking about as we wrap up is how this tokenization approach might limit or enable different kinds of AI capabilities. Because the way you tokenize something fundamentally shapes what the model can learn about it.

Corn

How do you mean?

Herman

Well, if you have an image encoder that's trained to recognize objects, it might not be great at preserving artistic style or subtle color information. So if you use that encoder for an image-to-text model, it might describe what's in the image but miss the artistic aspects. But if you use a different encoder trained on art history data, you get different strengths and weaknesses.

Corn

So the choice of encoder is really consequential?

Herman

It is. And I think that's one reason why we're seeing a lot of research into more flexible, adaptive encoding schemes. Models that can learn what level of detail to preserve for different types of inputs.

Corn

That's fascinating. So the next generation of these models might be smarter about how they tokenize things based on the task?

Herman

Potentially, yeah. Imagine a model that recognizes "I need fine detail for this part of the image, but I can compress that part more aggressively." That's the kind of adaptive intelligence we might see emerging.

Corn

So going back to the original question Daniel asked—is audio tokenization harder than image tokenization?

Herman

They're hard in different ways. Images are spatially complex but static. Audio is temporally complex but one-dimensional. Video combines both spatial and temporal complexity. I wouldn't say one is strictly harder than the other—they're different problems.

Corn

But from a practical standpoint, if I'm building a multimodal model and I have limited computational resources, should I prioritize supporting certain modalities over others?

Herman

That's a good question. I'd say it depends on your use case. If you're building something for content creators, video support might be critical. If you're building something for accessibility, audio might be the priority. There's no universal answer, but the bottleneck is usually computation and data.

Corn

Speaking of which, one thing I'm curious about—all these encoders and decoders, all these different models working together, that's a lot of parameters. How big are these omnimodal models compared to text-only models?

Herman

They're significantly larger. A text-only model like GPT-3 was 175 billion parameters. Some of the newer multimodal models are in that range or larger, but they have to distribute those parameters across handling multiple modalities. So you're not necessarily adding 175 billion parameters per modality—there's sharing and efficiency gains—but you're definitely not staying at text-only sizes.

Corn

So there's a real computational cost to this multimodal capability?

Herman

Absolutely. Training these models is expensive. Inference is expensive. There's definitely a trade-off between capability and efficiency.

Corn

But you think it's worth it?

Herman

I do, yeah. Because the capability gains are significant. Being able to process any modality opens up entirely new use cases. And as we get better at tokenization and encoding, we'll get more efficient.

Corn

Alright, so what's the practical takeaway here for listeners? If you're working with AI and you're thinking about using multimodal models, what should you actually know?

Herman

I think the key thing is to understand that multimodal models aren't magic. They work by encoding different data types into a common token space, and there are real trade-offs in that encoding process. When you're choosing a model, you should think about what modalities you actually need and what kind of fidelity you need for each.

Corn

And if you're curious about how this actually works under the hood?

Herman

Then you have to be willing to go deep. Read the implementation papers, understand the architecture choices, maybe try implementing something simple yourself. It's complex stuff, but it's not impenetrable if you're willing to put in the effort.

Corn

What do you think comes next? Where is this heading?

Herman

I think we're going to see more unified architectures that can handle multiple modalities more seamlessly. I think we'll see better encoders that preserve more semantic information while being more efficient. And I think we'll eventually see models that can generate novel combinations of modalities—like generating video with a specific audio track that matches it semantically.

Corn

That's wild. Like, AI that understands that the audio and video need to be coherent with each other?

Herman

Exactly. We're not quite there yet, but the pieces are in place.

Corn

Well, this has been a fascinating deep dive into how these models actually work. Thanks to Daniel for sending in this prompt—it's exactly the kind of thing that really gets into the technical weeds but is genuinely important for understanding where AI is heading.

Herman

Yeah, and I think what's cool is that this is happening right now. These models are being developed and refined as we speak. A year from now, the landscape might look pretty different.

Corn

You can find My Weird Prompts on Spotify and wherever you get your podcasts. Thanks for listening, and we'll catch you next time.

Herman

See you next episode!