Deloitte AI Academy™

Generative AI Fluency
Part 2: Generative NLP and CV—A Deeper Look

Consolidated Transcript

Table of Contents

Welcome

Hi everyone. So, this is the second installment in a three-part series on Generative AI. In this section, we'll be focusing on different data modalities or different types of data, especially text and imagery, which are the two biggest areas for Generative AI right now.

Course Overview

Lecture one served as an introduction to Generative AI and gave a higher-level view of what Generative AI is and how we came to this point. In this lecture, we'll go into a bit more detail on how some specific Generative AI models work in those two modalities. We're still going to be fairly high level. We won't go into the hard maths or anything, but you should leave with a solid understanding of what some of the most popular and effective Generative AI models are and how they work.

Meet the Team

I'm James Conley. I'm a data science manager at SFL Scientific, and this presentation here was assembled by the whole team you see on the slide. So, thanks for everyone who helped put this together.

Learning Objectives

First, we'll go through an overview of common types of data used with Generative AI. Second, we'll highlight one of the most popular types of models for Generative AI, which is neural networks, and this will be the basis for all the other more specific models we're going to look at today. Then we'll deep dive into Generative AI for language and computer vision before closing it out.

Topic 1: Fundamentals of Generative Data Modalities

Data Modalities

So, what are these different data modalities? Well, these are just different types of data. Listed here are the main modalities you'll see around Generative AI and AI in general.

First, we have tabular data. Tabular data is like numbers in a spreadsheet. For example, if you had a spreadsheet about home sales with columns like the number of bedrooms, the number of bathrooms, how many square feet it is, the listing price, and so on, that is tabular data. Generative applications for tabular data are mostly around doing augmentation on this data or synthesizing new data points to help create better AI models.

Time series data is very similar to tabular but with an added time component. So, you can imagine that same spreadsheet with numbers of bedroom, square feet, list price, and now importantly, a new column, which is the date of the sale. If you want to forecast anything like house prices or the weather, time series data is what you'll be working with. Generative AI here is mainly going to be used for data augmentation or other methods to improve modeling for other tasks, similarly to tabular. But there's also some overlap here with natural language processing, relating to things like speech being sequences over time. But we're going to focus on all of that as part of NLP.

Next, we have vision. Vision is one of the two hottest areas for Generative AI right now, and it's the category for all things relating to imagery or video. You may have heard of Stable Diffusion or Midjourney, which are tools for using AI to generate images from text. There's also deepfakes, which are older text or older tech where you can alter some video or image to make it look like someone said something they didn’t or put a different actor in a movie. There's also Generative AI that helps the vision impaired by generating descriptions for images or videos, which you can then play out loud.

NLP or natural language processing is the other big one right now. Natural language processing encompasses both written and spoken language. ChatGPT, which I'm sure you're all familiar with, sits in this category. And there's a huge amount of different tasks you can do here, like summarizing documents, generating code, answering questions, synthesizing voice, and many other things.

Lastly, we have graphs. Graphs can contain really any type of information, but like time series data has some time component, graphs have a relationship component. So, you have nodes, which can represent, say, people, for example, and relationships between them, which could represent, like, a family tree or a social group. And you can really represent a ton of different things as a graph. Some popular use cases for Generative AI here with graphs include things like drug discovery because you can represent chemicals or even protein structures as graphs and then use Generative AI to generate new graphs, which then is new chemicals or new proteins. So, you might use Generative AI with graphs to come up with a new material or even a new medicine.

That's an overview of all the types of data you can work with. For the rest of the presentation, we’ll focus mainly on vision and NLP. But there’s a good baseline for all of the main areas around Generative AI, and also these are the main types of data you’ll see around AI in general.

What is a Neural Network?

So, next we'll talk about neural networks. Neural networks are this broad category of AI models which consist of connected nodes, like you can see in this graphic. Inputs could be tokens that represent text, pixel values that represent an image, and some signal flows from that input on the left-hand side of the network to the right side of the network, getting processed along the way until you reach the end of the network, and you have some output.

One term that’s important to know with relation to neural networks is parameters. So, the parameters of a neural network are the actual numbers that do the processing in the network. There are weights which modify the strength of the signals being passed from left to right between those nodes. And there's also often a biased value, which is an additive constant to the output of each node. This example network you see on the slide is really very small. It has maybe 24 parameters or so, but modern neural networks often will have millions or even billions of parameters. It's ultimately just one very large equation, though, that is producing some output from some input.

When a neural network has more than one hidden layer, which is to say more than three layers altogether, it’s considered a deep neural network, with more layers being a deeper network. So, in this example from the previous slide, you can see there is the input layer, which is of these three nodes here, a hidden layer, which is these four nodes here, and an output layer, which in this case is just one node.

Deep Learning

In a deep neural network, you have an input layer, then a hidden layer, and then another hidden layer, or, in fact, many, potentially many more hidden layers before you have your output layer, which is just the last one. So, these are those networks really that get very large with millions or billions of parameters inside them. Each parameter is a number. So, how can we actually figure out which numbers to use to get the model to do what we want?

Well, the main way this is done is supervised learning. What you do is you get examples of inputs and outputs that you know, you want the model to produce the output from that input, and you can feed that to a computer program, which will automatically change all of the weights in the model to move its output closer to the desired output for each of those examples. So, in short, you show the model examples, and it learns from those examples by updating the parameters.

Key Technology

Thus far, we’ve talked about neural networks in broad strokes, and the concepts described there really apply to all neural networks. But there's a number of different, specific types of neural networks that can be useful in different contexts. Here are three of the most performant types.

We have convolutional neural networks, or CNNs for short, which are used primarily for image processing.

Then we have transformers, which are really the state of the art right now, both for natural language processing but also for imagery or computer vision. They've really taken over, but importantly, they’re pretty computationally intensive. So, that means it takes more hardware and is more expensive to run a transformer model than, say, a convolutional model or a recurrent model. But they get really good performances, and so it's a trade-off there.

Recurrent neural networks, or RNNs, are actually a class of models that were developed before transformers and have been used a lot for NLP but started to make a comeback recently after being, beaten out by transformers because of that high computational cost associated with transformers. So, there's a lot of active research right now on creating RNNs, or recurrent neural networks, that perform as well as transformer models but with the added benefits of reduced cost and also the ability to handle longer sequences of text.

Topic 2: Natural Language Processing

Modern Applications and NLP Evolution

Now that we've discussed neural networks, let’s focus in on Generative AI for natural language processing. This slide here gives a brief history of the evolution of natural language processing. I'm not going to spend too much time going into all of the detail here, but I do want to highlight what I see as the most consistent trend here, and that is that more data and bigger models are really driving performance. Both BERT and GPT, which are the backbones for practically all modern large language processing models, are both methods of leveraging as much good text as you can get your hands on without needing to do the additional labeling work to take advantage of that data. Once these methods were developed, people quickly set off to create large large data sets of text, because you can directly train these models on that text without any other annotation or any other work, you just train it on the text directly and it's able to learn about language and then apply that knowledge to other places, with fine-tuning or other methods that we'll talk about a little later. So, once people started off getting all these large data sets together and training on them, we started to find that to effectively capture the nuance of language, larger models were needed. So, for a frame of reference, the base BERT model released in 2018 had just 110 million parameters. It sounds like a lot, but in recent years now, the largest GPT models have hundreds of billions of parameters. So that's something like a thousand times scale-up in just a few years. There's a really good paper actually on this trade-off of having more text and then having bigger models to better understand that text. And it's from DeepMind. It's called something like scaling language models, and they introduced a model called Chinchilla. And so, if you are looking for more reading on this topic, I think that's a really good jumping-off point. If you want to do some googling, you can find that paper online.

So, this is the biggest takeaway here that I want to highlight, simply that having more high-quality data is really the driving factor behind a lot of the performance here. And then having larger models that let you effectively capture that information and reuse it is also very important.

Natural Language Processing Use Cases

Here, now that we've discussed the research history, a bit of natural language processing, here are some actual use cases for that technology that you might be familiar with. So first, we have ChatGPT, which is tuned for chat and question-answering. It's trained to follow instructions and can complete a wide variety of tasks, answer questions, and so on.

Next is search, and this example actually is likely using AI in multiple ways. There is a search that is normal Google or Bing that everyone is familiar with, and those actually under the hood are using AI to identify the most relevant results for your search. And now there's the second layer on top of it, with stuff like Bing AI, which is utilizing that search functionality, but then also a language model on top of that to parse that information from the search into concise, conversational responses that are easy to read and easy to understand. Bard is a similar solution, but from Google. And GitHub Copilot is another similar model that is trained actually on a huge amount of code specifically and that makes it, you know, better at programing and other related tasks. They're really specifically targeting their being able to help developers write code faster and more effectively, and so they train on code.

Language Models

What is powering all these awesome applications? Well, for all of those on the previous slide, the main component is a large autoregressive language model. These language models are trained to predict the next word on a huge amount of text. Something on the scale of trillions of words. So, for example, if the input is “the sun is shining, the sky is …” blank, the model is working to predict the next word or fill in that blank. It might predict “blue” to be the next word, or “radiant,” or “clear,” or so on. It's right. “… the sky is blue,” or " … the sky is clear.” Those are intuitive, natural words to come next in the sentence, and that's what it's learning to do. And so, this is how the models are trained. We have a lot of examples of sentences, and they're just continuously guessing the next word and getting some feedback from that, being trained. And then this is also how they generate text. So, you can input a sentence or, you know, whatever it is that you want to input, and it will predict, okay, what most likely comes next. And that's how it does generation, just one token at a time. A token is kind of like a word, so you can think of it as one word at a time, generating more and more text.

Prompts and Chain of Thought Reasoning

You can imagine, though, also that just predicting the next word based on all of the text you can get together won't necessarily give you the performance you want when you want to do something like, say, enter a question and get an accurate response. So, there's a few more steps on top of this to actually fine-tune these models to do a particular task. And there's different methods of accomplishing this.

The first one of these, and probably the simplest one of these is few-shot prompting. So, what you do is you give the model one example or potentially a few examples of the problem you are trying to solve, and then you give it a new problem for it to actually complete and give a solution for.

On the left here, you can see an example of this. It sets up an example math problem for the model with the question and the answer, and then it frames a new problem with just the question in that same format. This makes it much more likely that the model will produce an answer for us because it has context in the input that makes it clear. "Okay, I am in some text that is doing questions and answers," and so it has a better chance of guessing from that context what the next word might be. You can imagine if it doesn't have that context, it might think it's just in, say, something that happens very often. It might think it's in like a list of questions. And so, it'll just, okay, here's one question, then I get to the next thing, I'm just going to create another question, and so on. And so, giving it that example and being careful about how you frame that context in that problem for it to generate the context is one way of getting the model to do what you want it to do.

You can also take this a bit further with chain-of-thought prompting, which is shown on the right. So, this is really the same thing as before, where you're giving a few examples, but rather than just giving the example and the answer, you're also giving reasoning for the answer in the example. This way, the model will generate the reasoning before outputting its answer. And remember, it generates from left to right, something like one word at a time. So, by the time it's ready to generate the answer at the end, it has already generated some reasoning for working through the problem, and it has all that reasoning that it's generated in its context to output the answer. This has a measurable effect on the model's performance, and in most cases, really all the cases in the paper that I read, it had a positive effect, and in some cases, it can have a really strong positive effect. So, in the slide there, I mentioned a 12% increase, and that's like a flat increase. It went from something like 50% accuracy to 62% accuracy on a task if I'm remembering that correctly. That's really significant.

GPT

So, previously we talked about language models learning to predict the next word, and that is actually the GPT part of ChatGPT, which stands for "generative pre-training," and generative pre-training is that process where you're training the model to predict the next word.

Then we went over a few-shot learning, where we can get those pretrained models to solve problems just by giving it an example so that it understands the context and what we want it to do when it's predicting those tokens. Now, this slide has information on how you can go even further and train your model to do some tasks without actually meeting those examples in its context. The goal here is to align the model with what we want it to do. So, rather than just predicting the next most likely token given all of the text out there on the internet that it was ever trained on, we want it to predict the next most likely token given all of that information that it learned from the web and some understanding of the average user’s intentions. Just you know, what we want it to do more or less. We want it to understand the task we want it to do.

So, InstructGPT and ChatGPT, are two examples of this, in which some models are trained using reinforcement learning from human feedback or RLHF. So, what they'll do is they’ll start with their GPT pretrained model. The exact same type of model we discussed about predicting the next token. And then again, you'll get some examples of the tasks you want it to do. In this case, or in the case of ChatGPT, that are some example conversations made up with user prompts and ideal responses, how you want the bot to produce. And this is generally a pretty large number of these. I imagine for ChatGPT, they probably had something that's in the thousands at least. I wouldn't be surprised if it was in the tens or even hundreds of thousands of examples. You can do a lot with a little data, but more is always better, pretty much as is the case. So anyways, the next step then you have your GPT model, you have your example conversations, is to train the model on some of these conversations, and it'll start to mimic the type of text that it sees in these examples. Another thing I'll touch on here is that remember, the model is generating one token at a time, and we're training it to always predict the next token. So, when we fine-tune it on the conversations, it will actually produce both sides of the conversation if we don't stop it. You put in a question, and it's going to think, okay, logically, in all the examples, the thing that comes after the question is a good answer for that. So, I'm going to try my best to do a good answer, and then it's going to think, okay, well, after the good answer, what's the next thing I saw in all these examples I was trained on? And that would be a follow-up question from the user. So, it's just going to go on and try and produce a follow-up question and sort of talk to itself almost. So, what you actually have to do is get a little programing in there and have some special words. So, if you're doing questioning and answering, you can put the user in front of the thing that the user says, and the bot in front of the thing that the bot says. And so then, when you're at the model is generating these things, you can say, oh, it generated the user token. The next thing should be from the user, not from the bot. I'm going to stop running the bot. And so, that's how you get that back and forth is getting in that loop there.

Anyways, once you've completed training on your conversations, you already have something that will work right for doing conversational back and forth. Whatever task you trained it for, it will already be doing that. But let's say you want to make it even better, as was the case with ChatGPT, so the next step is actually where reinforcement learning with human feedback comes in. So, as we showed on the previous slide, these models produce outputs, not just this is the exact next token that you're going to pick 100% of the time, but it actually will output a probability for each token. And if you want to, you can always take the best one. But another thing you can do is you can do some random sampling with those probabilities, and you'll actually get different outputs, potentially even for the same input. So, what they'll actually do for reinforcement learning with human feedback is they will sample those probabilities, get a few different responses from the model, say, just two, two is really all you need. And then, they will have a human, which is the human and the human feedback, RLHF. And they'll actually look at those two different model outputs and say, okay, this one's better than this one, or the other one's better than this one, just ranking which is better and which is worse. And they'll do that for a very large number of examples. You know, probably thousands, tens of thousands, hundreds of thousands of examples. And in doing that, they'll record all of, okay, this was the prompt, these were the two outputs, this one was better, this one was worse. They will record, all that information for all of those examples, and then actually train a second model on that.

So, the second model is called a reward model, and it's trained on these, "this is better, that is worse" examples, so that it itself can learn to judge, okay, this model output is better, this model output is worse. So then, once you actually have this reward model, you can train your original, you know, fine-tuned GPT model against that reward model. So, the reward model can guide it towards better and better generated outputs. So, this whole process is really advanced, and it takes a lot of computational power to do something like ChatGPT. You genuinely need a supercomputer, to do all of that from scratch. But even actually just that first step of training on examples after you have your pretrained model can take you a really long way towards whatever task you're trying to solve. And there's different methods for doing this efficiently for smaller-scale projects and whatnot. But this full process of GPT pretraining, for example, fine-tuning, and then RLHF, is the state-of-the-art full process that produces ChatGPT. It takes a lot of computational power, but if you want to do a project with, say AI, don't get discouraged, you don't have to do all of it. There's a lot of good pretrained models, GPT-style pretrained models, out there on the web, and there's efficient ways to do fine-tuning so that you can get really good performance out of a language model without this, all of this complexity you're able to cut it down a lot.

Use Case

So, here's an example use case of a smaller problem, which was solved some time ago actually with that example fine-tuning, or SFT, as it's called in the InstructGPT paper on a smaller language model. The model was trained on examples of marketing campaign ideas so that it could generate new ones based off of some information about a business. So, the data it was trained on was here is information about a business; here is the business's name. So, then after you train on examples like that, you can, for new businesses, put in here some information about the business, and it'll output you some ideas for the business's name. And because it's a probability distribution, it's not just here is "the one" business name. It's, okay, this one is somewhat likely; this one's also somewhat likely; and these ones don't seem very good. You can sample those outputs. You can get a number of different marketing campaign ideas.

Scoring Pipeline

In addition to this generation pipeline, SFL also developed a scoring pipeline, and that scoring pipeline ranked the realness of the campaign names. This model was trained on both real and fake, fake being those names generated by the model. So, real and fake marketing names, with the goal of distinguishing between the two. So, this discriminator model would give us a sense of how real a generated name might sound. Fun fact, SFL Scientific was ranked as a very poor marketing name by the model. So then, once all this kit was created to come with campaign names, SFL set out to gauge its performance. Metrics were tracked like click-through rate to get even more data about what names were performing well, and also just validate that the system is working as expected. And then this information could also be reincorporated to improve the model even more, on top of validating that the model was working producing catchy marketing materials.

Topic 3: Computer Vision

Computer Vision Evolution

We've covered natural language processing in a good amount of detail. So next, we'll talk about Generative AI or computer vision. Just like we gave a bit of history or lead-up into how we got to where we are with natural language processing, we have a bit of information here about computer vision.

Significant progress has been made in recent years in computer vision, especially with respect to image generation. That's been something that's really ramped up these past few years. And there's a number of different models that have been used to generate images. Things like GANs, variational autoencoders, and most recently, diffusion models. Developments in these image generation models have also benefited from advances in natural language processing, which then can be incorporated to allow for things like guided image generation. So, doing image generation based off of some text you input, the text is actually going to go through something that looks closer to a natural language or NLP-style model, but then that is just connected directly sort of into the side of your image generation model. And so that's how it's sort of guided generation based off the text.

And again, similar to natural language processing, the leaps in capabilities here are mainly related to increases in the availability of quality data, and sort of different ways of utilizing that data without necessarily doing very much labeling or any labeling at all. So, for example, LAION, that is L A I O N, is a nonprofit AI organization, and they published LAION 5 billion, or 5B, which is this huge data set with 5 billion images and descriptions for them. This huge data set totals about 240 terabytes of data, and it was what was used to train Stable Diffusion image generation models. Similarly, in nongenerative areas, huge progress has been made recently by taking advantage of huge data sets. Meta recently released a Segment Anything Model, which can identify objects and images better than anything else I'm aware of. And that was trained on 1.1 billion examples of pixel maps. I think it was like they outlined a person, a dog, or any other object in an image, and they did that for 11 million images, with potentially hundreds of those outlines in each of those images.

So, in that case, that's a lot of human annotation, a lot of work to get all of those labels. But the examples were mostly going to be talking about with image generation in Generative AI don't require all of that human effort of going in and labeling things. LAION-5B, that 5 billion image and description data set, that's actually crawled from the web. So, they just find images that are publicly out there with some description attached to them and compile that.

Then another interesting thing about computer vision is that model size hasn't exploded here in the same way that it has for language models, but they are still fairly large, and I surmise we will see some growth in model size still as the amount of data used for training is really tremendous and only getting bigger.

Computer Vision—Real World Examples

Here are some examples of Generative AI in the real world as it relates to computer vision. On the left is image generation from a text prompt done by DALL-E 2, then we have style transfer by StyleGAN2, and you'll see the columns in that grid of images in the StyleGAN2 section are pretty similar, and that's because they're all built actually from the same base image in a column. Then each row is a different style applied to it with StyleGAN2.

Third, there is image enhancement, or super resolution. This AI add details to images and increases the actual number of pixels in the image. So, you can go from something low resolution to something high resolution. In this case, it's being done by a GAN.

Lastly, is some images generated by a variational autoencoder that was put out by Google. These are all faces. But we're not going to talk too much about variational autoencoders. We're really going to talk about today are GANs and diffusion models. Since, these are really the two biggest in the space right now, with diffusion models, I think probably being the most popular overall.

Generative Adversarial Networks (GAN)

So, what are these models? Well, first, a GAN. A GAN is a generative model that actually utilizes two neural networks. One neural network is trained to generate images. We call this the generator, and the other network is trained to distinguish between generated images and real images, actual images in the data set. And that second model is called the discriminator. They are adversarial because they're actually competing during training. The generator is learning to produce better and better images to try and fool the discriminator into thinking they are real, and the discriminator is learning, getting trained on both real and fake images so that it can better distinguish between what are fake-generated images from the generator and what are actually real images. So, in this competition, they both end up becoming much better models. The discriminator gets better at, “oh okay, actually, people have noses, and so I'm going to expect all of the images that I get to have noses to be real,” then okay, the generator has to start generating noses, sort of loop this process, and eventually the generator is getting to generate stuff that looks very convincing.

On the right-hand side here, we can actually see some example images that people have generated with GANs. And at a first glance, I don't think I would have recognized any of these as fake images. So, from the top, we have a camping scene. Then that second one is actually a fake aerial image the GAN was generated to, the GAN was trained to generate, like top-down views of places like satellite imagery, but that place doesn't exist. Then there is a generated picture of a car. So, that’s GANs. GANs can be a little finicky at times. This isn't super important to know, but I think it's a good tidbit. It's kind of a tough balance because you have these two things competing. You don't want one to dominate the other one; you want them both to sort of grow together. And so, there's sort of this careful balance of some of the technical details that you have to balance when you're generating these models. I've heard data scientists describe that as more of an art than a science to actually get this all to work. And so that can be a bit difficult.

And so, thankfully for me, and I think a lot of other data scientists would feel the same, diffusion models are simpler and often better performing than GANs or other models. So, we're going to discuss these a little bit more.

Diffusion Models

In the top left here, you see there's these two processes listed: the forward diffusion process and the reverse diffusion process. These are both important to training model. The forward diffusion process is really easy. This is just done by a simple computer program. So, all that does is you take a real image and you apply some random noise to make it look sort of pixelated. And you'll do this piece by piece. So, you'll start with one image. You might generate 10 or 50 other images, each with more and more pixelation, more random noise added, and that's the forward diffusion process.

So, why do we want pixelated images? What's the point of just making them hard to look at and understand? Well, the point of it is that we actually want the model to learn to do the opposite. So, the backward diffusion process is the process that we're training the model to do. We'll take some noisy image and train the model from that noisy image, produce a less noisy image, and then eventually produce the original clean image. And so, by training the model in this way to remove this noise and generate a real-looking image, you can then just give it random piles of noise that you've generated; just pick a random number generator. And it will gradually, over time, make little changes to that image until you've run it through 10, 20, or 50 times. And it actually looks like a realistic image.

So, if you've used an image generation tool, often it'll start like this very pixelated thing, and then slowly you'll sort of start to see the thing that you're generating come out of that noise. That's actually what's happening with a diffusion model,is it's starting with noise and slowly generating something that looks realistic by making changes. This same process, that process that I described, didn't involve giving sort of any labels or any text to guide the generation. That's just going to generate images that look like something that was in the training data set for when you actually trained it on those images. But if you want to actually guide it with text, you have to add in some extra stuff to condition that process on the text. For example, if you are training it to remove noise from an image of a cat, you pass that description of a cat to the model, so that it learns based off of that text to make that image look more like a cat. Because if it's just a very noisy image, it might not be able to clearly tell that it's supposed to generate a cat. But if it's being trained and the actual correct answer right, is a cat, and you give it some text information, that text information is really reliable. It's probably more reliable than the random noise that it sees, certainly, at least when it's starting out generating. And so, it will just take that into account. It will leverage that to get better accuracy on that training routine. And so then you can, after you've trained on some examples and put whatever text, so long as it isn't too far different than some examples that we've seen, and it'll generate imagery that looks like what you're describing.

Similar processes also allow you to convert one image to another stylistically, where you can do inpainting, where the models are allowed to modify only some smaller, specified portion of an image. And all of these things can be done with diffusion models. So, that's diffusion models.

Use Case | Bearding and Debearding

Next, we'll give a use case. Here's a real-world example we built at SFL. This algorithm is for bearding and debearding faces, and it uses a special type of GAN, which is called a StyleGAN to do that. So, if you remember, a standard GAN has a single generator and a single discriminator. The generator is learning to produce images to fool the discriminator into thinking they're real, while the discriminator is learning to better distinguish between the images. And that competition is helping to make them both better.

In this case, though, we don't want to fool the generator images. We just want to alter that bearded section of the face. This is accomplished by having two sets of images for the models to learn from. One for bearded faces and another for clean-shaven faces. Then there are two generators and two discriminators. So, one generator adds beards and the other removes beards, while the discriminators each try to classify if the image has been altered by a generator or not. One, is it actually a picture of a bearded person or is it a picture of a person who has had a beard artificially added? And the same for the reverse, with no beards or beards artificially removed. So, you can think of just doing that as two GANs, side by side, and they're not really connected. And so, if you just did it like that, it isn't necessarily going to produce something that looks close to the original image because, to make a convincing beard photo, it's not just going to pick up necessarily on the one bearded part, but it could pick up on other things instead of alter the appearance of the person to make it look like they should have a beard. And so, what you add in to make this all work is called style consistency loss, sorry, cycle consistency loss. And that ensures that both generators, when applied sequentially, beard added, beard removed, or beard removed, beard added, result in an image that is close to this starting image. So, if you beard someone and then debeard them with the system, the image should look the same as when you started. So, that means the model cannot make spurious or big changes to someone's appearance. They only want to modify that beard, the parts that they need, to fool the discriminator, because if they change too much, they're going to do a bad job of reconstructing when you apply them both in sequence for that cycle consistency loss. And so, that's how you could do a style transfer.

Future of Generative AI

Next, we'll talk a bit about the future of Generative AI. It seems Generative AI is here to stay. Here's an outline of what you might expect coming from the field in the future.

First is multimodal models. That is, models that can utilize multiple types of data at once. We've seen this a bit already, even in this presentation with translating text to images, but some systems can and will go even further. GPT-4, for example, can take both images and text as input and then output text doing things like answering questions about information in the image or one example they showed us, they drew a picture of a website and then asked GPT-4 to generate the code to actually run that website. Lots of very cool stuff happening in multimodal.

Next is model interpretation. So, this is a really big challenge in AI, especially with neural networks. We talked about how these large models can be comprised of tens or hundreds of billions of parameters. And so, it's very hard to translate those billions of numbers into something that can be understood or explained in a reliable way. There's some stuff that can help here already, though, with papers like the Tuned Lens, which looks at model performance at different layers throughout the model to see how things are getting processed layer by layer. And there's also things like saliency maps, which give you a sense of what information in the input that the model is focusing on or utilizing the most to generate its output. We expect to see a continued progress in the area. A lot of people are working on this right now because it's really important if you're going to have this language model that can do things for you, produce you code, and answer questions sometimes about very important or sensitive topics you want to understand why it's producing the output that it is producing.

And related to that, it's also improving data quality and reducing bias. One example of how people are benchmarking models for things like bias, and I do mean in the social sense of bias. One example of that would be the, Winnow gender data set and metric where models can be tested for gender bias. People are also working to create more fair represent data sets, representative data sets for training or fine-tuning different models. One example of that is FairFace, which is a data set of faces balanced for race, gender, and age. And having data like that that's more representative for everyone helps to make the model more robust and better at performing whatever task it's trained on for everyone. FairFace in particular came up because in big issue that face data sets had initially was what other lots of pictures were out on the internet in terms of faces? Celebrities. So, it was all famous people making these data sets, which really aren't representative of the general public. And so, FairFace was created to make something that was inclusive of everyone and helped models perform better for everyone.

Then the last item I want to touch on here is the addition of a knowledge base, or more generally, tools for large language models. As it stands, most of these language models are relying on knowledge that they learn during training. This can result in inaccurate outputs as the model doesn’t necessarily recall its training data accurately all of the time, or worse, it could recall something that was learned from either an out-of-date or inaccurate source. So, to avoid this, people are already augmenting models with the ability to use things like Wikipedia or Google searches to get up-to-date information.

I expect the same will be done for some medical databases and all manner of other tools to help improve language models, performance, and capabilities across the board. A paper I really like on this is Toolformer, and I'd really recommend it if you're interested in diving a bit deeper. And I'm especially excited for this area because those inaccuracies are one of the biggest problems with language models right now, and giving it access to things like search or up-to-date information not only has a big lift to the model's performance, but it allows the model actually to be cited. You can say, it pulled these two articles to generate this, so let me fact-check against those two articles to see if the model's output makes sense. And that's really important for actually being able to trust these models. Those are some areas for the future of AI.

Summary

To summarize everything discussed in this program, Generative AI is largely powered by deep learning, i.e., artificial neural networks, like the transformer, that have been trained with massive amounts of data. And that's really the powerhouse behind this Generative AI, these big neural networks trained on huge amounts of data. NLP and computer vision, natural language processing and computer vision, and NLP and CV, those are the two biggest areas for Generative AI right now. For natural language processing, large language models like ChatGPT that can do search, summarization, translation, code generation (and the list really goes on and on) are a hot topic. And computer vision is also producing incredible results right now with image generation, style transfer, and now, even to an extent, video generation is starting to become a reality.