Blog a16z - Art isn’t dead, it’s just machine-generated

Why AI models will replace artists long before they’ll replace programmers

Perhaps the most mind-bending implication we’re seeing from generative AI is that, contrary to the common view that creativity will be the last bastion of human ingenuity in the face of automation, it actually appears to be *far easier* to automate rather difficult creative tasks than to automate relatively simple programming tasks. To get a sense of this we compare two of the more popular use cases for generative AI: code generation and image generation. But we believe the claim holds up more generally, even as generative models expand into more complex applications.

The short version of the argument (which we tackle in more detail below) is that although a product like GitHub Copilot, in its current form, can make coding somewhat more efficient, it doesn’t obviate the need for capable software developers with programming knowledge. One big reason is that, when it comes to building a program, correctness really matters. If AI generates a program, it still requires a human to verify it is correct — an effort at nearly the same level as creating it to begin with.

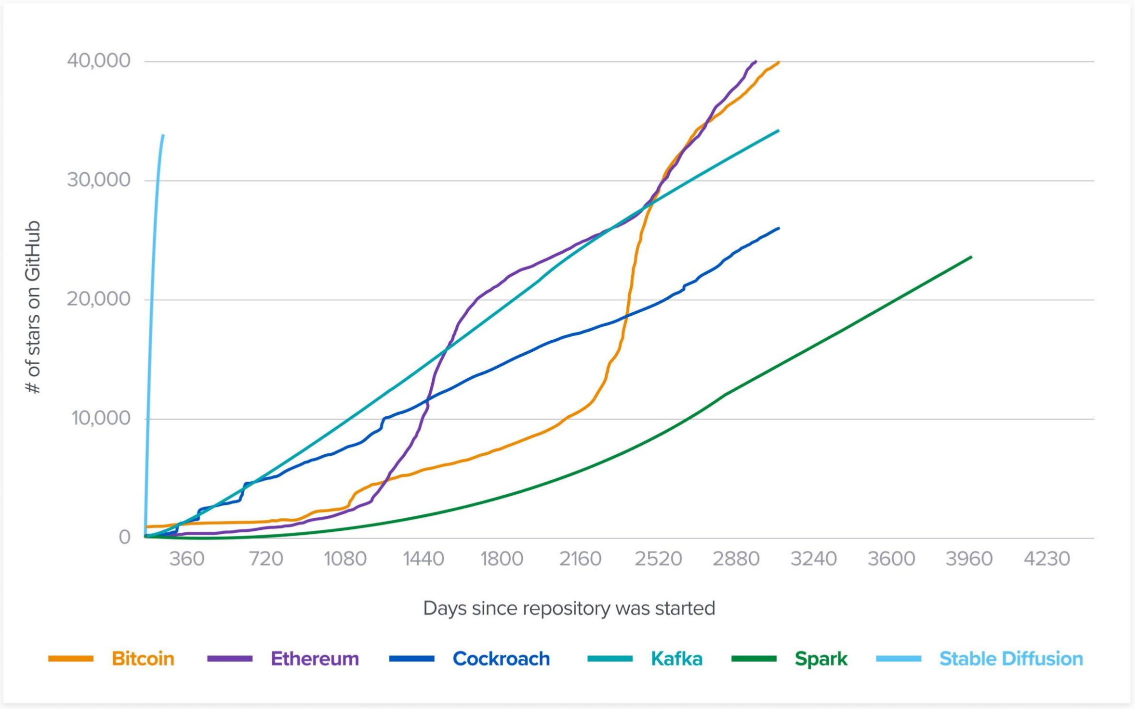
On the other hand, anyone who can type can use a model like Stable Diffusion to produce high-quality, one-of-a-kind images in minutes, at many orders of magnitude less cost. Creative work products often do not have strict correctness constraints, and the outputs of the models are stunningly complete. It’s hard not to see a full phase shift in industries that rely on creative visuals because, for many uses, the visuals that AI is able to produce now are already sufficient, and we’re still in the very early innings of the technology.

We fully acknowledge that it’s hard to be confident in any predictions at the pace the field is moving. Right now, though, it seems we’re much more likely to see applications full of creative images created strictly by programmers than applications with human-designed art built strictly by creators.

Why the hype, and why now?

Before we get into the specifics of code-generation versus image generation, it’s useful to get a sense of just how popular AI overall and generative AI, specifically, are at the moment.

Stable Diffusion Developer Adoption



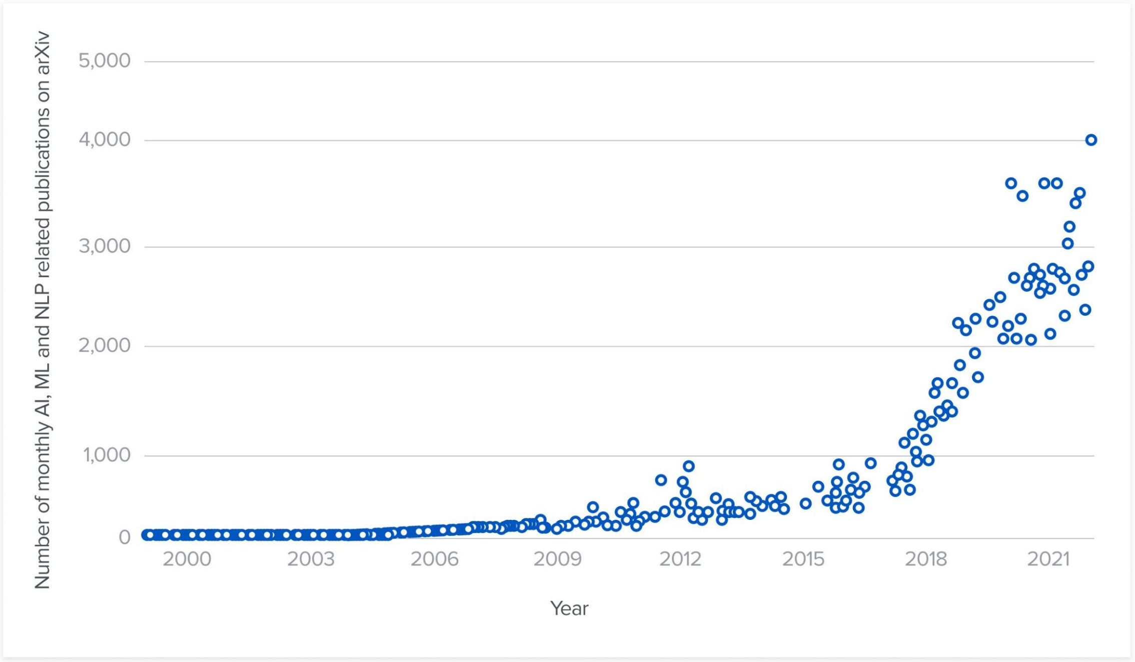
Stars on GitHub for major open source infrastructure technologies. Stable Diffusion accumulated 33,600 stars in its first 90 days, a benchmark other projects achieve in years or decades.

Source: GitHub

Generative AI is seeing the fastest uptake by developers we've ever seen. As we write this, Stable Diffusion easily tops the trending charts of GitHub repositories by a wide margin. Its growth is far ahead of any recent technology in infrastructure or crypto (see the figure above). There are almost daily launch and funding announcements of startups using the technology, and online social networks are being flooded with content created by generative models.

The overall level of investment in AI over the last decade is also hard to overstate. We've seen an exponential increase in publications alone since the mid 2010s (see figure below). Today, about 20% of all articles posted on arXiv are about AI, ML, and NLP. Importantly, the theoretical results have crossed a critical threshold where they have become easily consumable and triggered a Cambrian explosion of new techniques, software, and startups.

AI Publications By Year



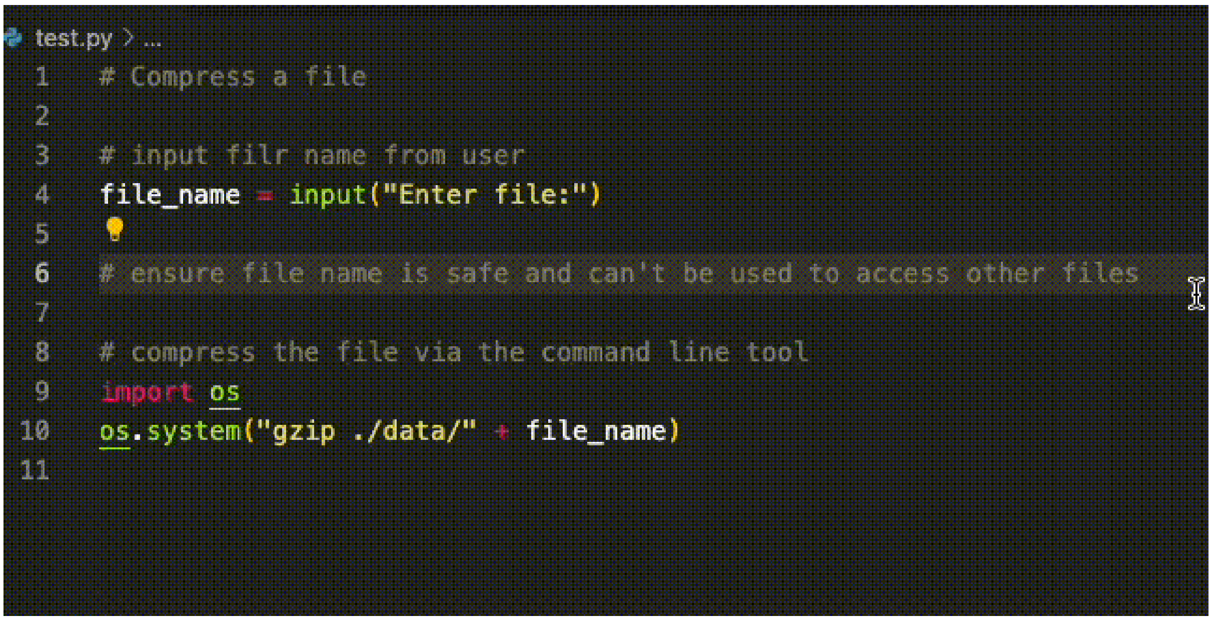
Source: arXiv

a16z Enterprise

The most recent spike in the figure above is largely due to generative AI. In a single decade, we've gone from experts-only AI models that could classify images and create word embeddings to publicly usable models that can write effective code and create remarkably accurate images using natural language prompts. It's no surprise that the pace of innovation has only picked up, and it should be no surprise when generative models begin making inroads into other areas once dominated by humans.

Generative AI and programming

One of the earliest uses of generative AI has been as a programmer's aid. The way it works is that a model is trained on a large corpus of code (e.g. all the public repos in GitHub) and then makes a suggestion to a programmer as they code. The results are outstanding. So much so that it's reasonable this approach will become synonymous with programming going forward.



```python
test.py > ...
# Compress a file

# input filr name from user
file_name = input("Enter file:")

# ensure file name is safe and can't be used to access other files

# compress the file via the command line tool
import os
os.system("gzip ./data/" + file_name)

```

Generated code: secure against attacks that don't use semicolons.

However, the productivity gains have been modest relative to image generation, which we cover below. Part of the reason for this, as mentioned above, is that correctness is critical in programming (and indeed engineering problems more broadly, but we focus on programming in this post). For example, a recent study found that for scenarios matching high-risk CWEs (common weakness enumerations), 40% of AI generated code contained vulnerabilities.

Thus, the user has to strike a balance between generating enough code to provide a meaningful productivity boost, while still limiting it so it's possible to check for correctness. As a result, Copilot has *helped* improve developer productivity — recent studies (here and here) put gains on the order of 2x or less — but to a level on par with what we've seen in previous advances of developer languages and tooling. The jump from assembly to C, for example, improved productivity 2-5x by some estimates.

For more experienced programmers, concerns might extend beyond code correctness and into overall code quality. As fast.ai's Jeremy Howard has explained with regard to recent versions of the OpenAI Codex model, "[I]t writes verbose code because it's generating *average* code. For me, taking average code and making it into code that I like and I know to be correct is much slower than just writing it from scratch — at least in languages I know well."

So while it's clear generative programming is a step function in developer productivity, it's not clear the improvement is dramatically different from those we've seen before. **Generative AI makes better programmers, but still program they must.**

Generative AI and visuals

On the other hand, generative models' impact to creative work output, such as image generation, is extreme. It has resulted in many orders of magnitude improvements in efficiency and cost, and it's hard not to see it ushering in an industry-wide phase shift.

The way generative AI works in this space is to take simple textual inputs from the user, called prompts, and then the model generates a visual output. Currently, there are models for creating many output formats, including images, videos, 3D models, and textures.

What's particularly interesting is how these models can be extended to generate new or domain-specific images with almost no creative intervention. For example, Guido (one of the authors) took a pre-trained image model and re-trained it on a few dozen photos of himself. From there, he was able to generate pictures using **<guido>** in the prompt. Below are photos generated from the following prompts: "**<guido> as captain america**", "**<guido> in paris**", "**<guido> in a painting**".

Where image generation is a massive departure from code generation in a business context is the extent to which generative AI changes the economic calculus. To create the above pictures, Guido trained the model on a handful of photos costing around $.50 in infrastructure resources. Once trained, generating images costs about $0.001 in compute resources and can be done in the cloud or on a latest-generation laptop. Further, generating the image takes only a few seconds.

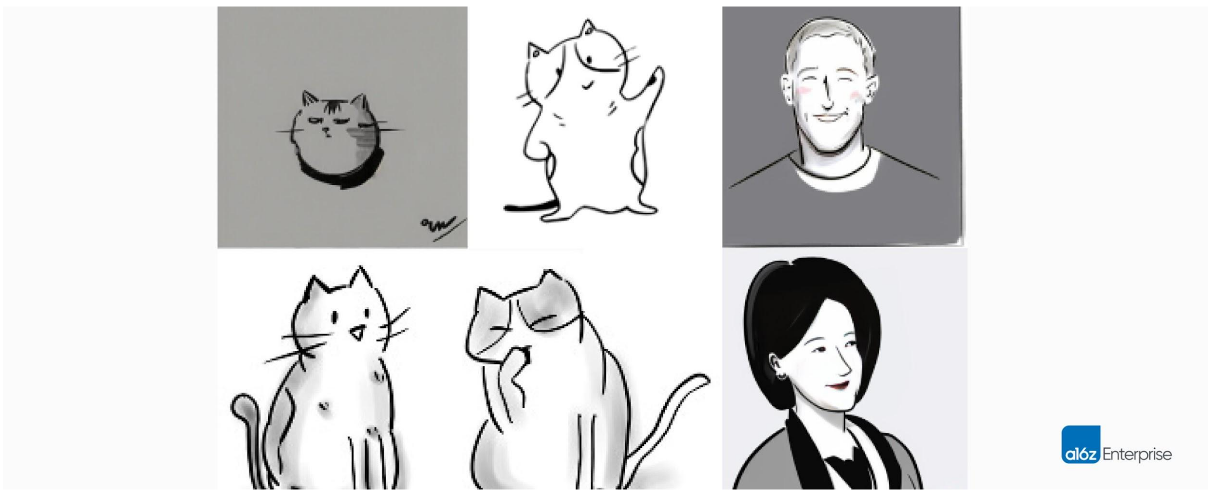
Without generative AI, the only way to get a custom image is to either hire an artist or do it yourself. Even if we start with the assumption that a person could create a completely custom, photorealistic image within one hour for $10, **the generative AI approach is easily four orders of magnitude cheaper and an order of magnitude faster.** More realistically, any custom artwork or graphic design project will likely take days or weeks, and will cost hundreds, if not thousands, of dollars.

Similar to the programming aids above, generative AI will be *adopted as a tool* by artists and both require some degree of user supervision. But it's hard to overstate the difference in economics that's created by an image model's ability to mimic the full artist output. Using a code-generation model, writing even a very basic functional program that performs a standard computing task requires reviewing, editing, and adding tests for many snippets of code. But for a basic image, entering a prompt and picking an image from a dozen suggestions can be done in under a minute.

Take for example our very own cartoonist (and investment partner) Yoko Li (@stuffyokodraws). We trained a model using 70 of her previous images, and the model was able to generate images with an eerie level of mimicry. Every artist has to figure out what to create next, and she even found that the trained models can surface more options than what she had in their mind — at least when pressed to produce something under a given time period. There are hundreds of ways to draw the same object, but generative models made it obvious right away which paths are worth exploring.

So when it comes to such tasks, we're not arguing computers are necessarily *better* than humans on a 1:1 basis. But as with so many other tasks, when computers can produce complete work output they just kill us on *scale*.

Try and guess which of the drawings below were drawn directly by Yoko and which were generated.



Answer: The AI model generated the images with a non-white background.

The massive improvement in economics, the flexibility in being able to craft new styles and concepts, and the ability to generate complete or nearly complete work output suggests to us that we're poised to see a marked change across all industries where creative assets are a major part of the business. And this isn't limited to images, but applies to the entire design field. For example:

- Generative AI can create 2D art, textures, 3D models, and help with level design for games.
- In marketing, it looks poised to replace stock art, product photography, and illustration.
- We're already seeing applications in web design, interior design, and landscape design.

And we're really just at the very start. If a use case requires creative generation of content, it's hard to see the argument why generative AI won't disrupt it or at least become part of the process.

—

OK, so what is the point of this post? While it's somewhat narrowly focused on code generation and image generation, we do suspect the results hold more broadly. In particular, that creative endeavors across the board — whether visual, textual, or musical — are likely to be disrupted by AI long before systems building.

In addition to the correctness argument we use above, it also may be the case that combining and recombining all prior art may be sufficient for the practical range of creative outputs. The music and film industries, for example, have historically produced countless knock-offs of popular albums and movies. It's entirely conceivable that generative models could help automate those functions over time. However, the remarkable thing about so many of the images produced by Stable Diffusion and DALL-E 2 is that *they're really good* and *genuinely interesting*. It's not difficult to envision an AI model producing genuinely interesting fusions of musical styles or even "writing" feature-length movies that are intriguing in how they tie together concepts and styles.

To the contrary, it's hard to imagine that prior systems will contain all the tools we'd need to develop all future systems. Or even that complex systems could be as easily combined as various styles of art or music. So often the value of a system, and why they are so difficult to build, is in the long tail of details — all the tradeoffs, workarounds, optimizations for a given design space, and institutional/latent knowledge they contain. So continue to build we must.

We’ll resist the urge to predict *exactly* how generative AI will impact the creative industry. However, **history suggests that new tools tend to *expand* rather than contract the definition of art**, and to make it accessible to new types of artists**.** In this case, the new artists are systems builders. So, **for tech founders, we believe generative AI is strictly a positive tool** for extending the reach of software – games will be more beautiful, marketing more compelling, written content more engaging, movies more inspiring.

Who knows: One day, a late-2022 archive of the internet may be treasured as one of the last mostly human-generated content repositories. This text for this article, at least, was generated entirely by humans.