**What is Artificial Intelligence?**

Artificial Intelligence is a system with increasingly self-learning capabilities that can supplement human cognition and activities. It does this by understanding the environment, solving human problems, and performing human tasks.

For example, [Google Maps uses AI](https://blog.google/products/maps/google-maps-101-how-ai-helps-predict-traffic-and-determine-routes/) to monitor traffic patterns and predict the best routes for you to take. Netflix also uses machine learning (a subset of AI) to power its recommendation algorithm, which suggests shows suited to your preferences.

AI systems like these operate by feeding data into intelligent algorithms that can learn and improve by analyzing patterns within the data. And the more data these algorithms receive, the better they get at understanding the environment and predicting patterns. These systems also can be trained to perform both simple and complex tasks, depending on pattern matching algorithms.

**Will AI Take Over Our Jobs?**

There’s a looming fear that AI will take over our jobs … and then the world. And while this makes for a great movie plot, the reality isn’t so scary. Human testing assisted by AI is still the best practice and will remain that way for the foreseeable future.

The Chief Scientist at test.ai, Tariq King, recently spoke about the future of AI in “Lighting Strikes the Keynotes,” at the 2022 STAREAST Software Testing conference. King pinpointed our biggest problem as being an abundance of bad software that people no longer trust. And with all this poorly produced software, there’s concern that AI could be an efficient way to create more of it.

But King went on to discuss a promising alternate reality: a world in which an abundance of good software is produced, and AI is kept stable by testers. Once the “AI revolution” occurs, we can build higher-quality software and AI systems and use that AI technology to enable trusted software. And through all of this, a new subset of testers will emerge as the heroes who oversee and support these improved AI systems.

So, while the future of testing will begin to change, our world isn’t becoming a sci-fi movie just yet.

**How AI Can Optimize Your Testing**

Software testing has evolved rapidly in the past few decades. It began with manual testing, moved to early automated testing, agile testing, and then continuous testing. And now, it’s reached AI-augmented [autonomous testing](https://smartbear.com/resources/ebooks/six-stages-from-manual-to-autonomous-testing/).

Achieving autonomous testing may seem like an impossible goal, but you can start implementing AI features that will create the foundation for it. AI can optimize your testing by accelerating test creation, expanding test coverage, and reducing test maintenance. But what does this look like? Let’s see some examples.

Object recognition is a form of intelligent design, which recognizes new objects and updates to add to the DOM without manual effort. There’s also framework generation that automatically scans your application to recommend a test framework.

Business process automation is an example of intelligent test execution, which helps to automate business workflows for end-to-end testing. Finally, self-healing is a form of intelligent test maintenance, which dynamically updates your test suite when your application changes or evolves.

**SmartBear Examples of AI in Testing**

TestComplete, SmartBear’s UI test automation tool, has an AI-powered feature called the [Intelligent Quality Add-on](https://smartbear.com/product/testcomplete/features/intelligent-quality/). It has intelligent capabilities such as self-healing tests, optical character recognition (OCR), and ML-based visual grid recognition.

These self-healing tests can automatically identify unexpected errors due to dynamic properties and recommend a better alternative. This prevents tests from failing, but more importantly, saves you time you would’ve spent reviewing these errors. OCR also helps you interact with content on complex applications by identifying objects based on text contents instead of their properties.

You also can be more efficient with ML-based visual grid recognition, which helps you validate data against complex objects like grids and tables. It does this by recognizing data within tables and verifying its contents against a baseline copy.

In the future, TestComplete seeks to introduce a higher level of AI to help you achieve autonomous testing. It will use AI to generate test cases, run tests, and modify test cases (even if the UI changes), all without human involvement.

**The Future Has Arrived**

AI technology still has a ways to go, but it’s already starting to make our lives easier from Netflix recommendations to software testing. And if we work to improve upon and utilize AI technology, things will only get better. Now start to embrace AI and be the key to shaping the next phase of test automation.

### 1. Do visual, automated validation UI testing

What kinds of patterns can ML recognize? One that is becoming more and more popular is image-based testing using automated visual validation tools.

"Visual testing is a quality assurance activity that is meant to verify that the UI appears correctly to users," explained [Adam Carmi](https://twitter.com/carmiadam), co-founder and CTO of Applitools, a dev-tools vendor. Many people confuse that with traditional, functional [testing tools](https://content.microfocus.com/software-test-automation-tb/buyers-guide-software-test-automation%20?utm_source=techbeacon&utm_medium=techbeacon&utm_campaign=7014J000000dVA4QAM), which were designed to help you test the functionality of your application through the UI.

With visual testing, "we want to make sure that the UI itself looks right to the user and that each UI element appears in the right color, shape, position, and size," Carmi said. "We also want to ensure that it doesn't hide or overlap any other UI elements."

In fact, he added, many of these types of tests are so difficult to automate that they end up being manual tests. This makes them a perfect fit for AI testing.

By using ML-based visual validation tools, you can find differences that human testers would most likely miss.

This has already changed the way I do automation testing. I can create a simple machine learning test that automatically detects all the visual bugs in my software. This helps validate the visual correctness of the application without me having to implicitly assert what I want it to check. Pretty cool!

### 2. Testing APIs

Another ML change that affects how you do automation is the absence of a user interface to automate. Much of today's testing is back-end-related, not front-end-focused.

In fact, in her [TestTalks interview](https://joecolantonio.com/testtalks/194-reality-testing-artificial-world-angie-jones/" \t "_blank), "The Reality of Testing in an Artificial World," [Angie Jones](https://twitter.com/techgirl1908), an automation engineer at Twitter, mentioned that much of her recent work has relied heavily on API test automation to help her ML testing efforts.

Jones went on to explain that in her testing automation, she focused on the machine learning algorithms. "And so the programming that I had to do was a lot different as well. … I had to do a lot of analytics within my test scripts, and I had to do a lot of API calls."

### 3. Running more automated tests that matter

How many times have you run your entire test suite due to a very small change in your application that you couldn't trace?

Not very strategic, is it? If you're doing continuous integration and continuous testing, you're probably already generating a wealth of data from your test runs. But who has time to go through it all to search for common patterns over time?

Wouldn't it be great if you could answer the classic testing question, "If I've made a change in this piece of code, what’s the minimum number of tests I should be able to run in order to figure out whether or not this change is good or bad?"

Many companies are using AI tools that do just that. Using ML, they can tell you with precision what the smallest number of tests is to test the piece of changed code.

The tools can also analyze your current test coverage and flag areas that have little coverage, or point out areas in your application that are at risk.

[Geoff Meyer](https://twitter.com/geoffrey_meyer), a test engineer at Dell EMC, will talk about this in his upcoming session at the AI Summit Guild. He will tell the story of how his team members found themselves caught in the test-automation trap: They were unable to complete the test-failure triage from a preceding automated test run before the next testable build was released.

What they needed was insight into the pile of failures to determine which were new and which were duplicates. Their solution was to implement an ML algorithm that established a "fingerprint" of test case failures by correlating them with system and debug logs, so the algorithm could predict which failures were duplicates.

Once armed with this information, the team could focus its efforts on new test failures and come back to the others as time permitted, or not at all. "This is a really good example of a smart assistant enabling precision testing," Meyer said.

### 4. Spidering AI

The most popular AI automation area right now is using machine learning to automatically write tests for your application by spidering.

For example, you just need to point some of the newer AI/ML tools at your web app to automatically begin crawling the application.

As the tool is crawling, it also collects data having to do with features by taking screenshots, downloading the HTML of every page, measuring load times, and so forth. And it continues to run the same steps again and again.

So over time, it's building up a dataset and training your ML models for what the expected patterns of your application are.

When the tool runs, it compares its current state to all the known patterns it has already learned. If there is a deviation (for instance, a page that usually doesn't have JavaScript errors but now does), a visual difference, or a problem of running slower than average, the tool will flag that as a potential issue.

Some of these differences might be valid. For example, say there was a valid new UI change. In that case, a human with domain knowledge of the application still needs to go in and validate whether or not the issue(s) flagged by the ML algorithms are really bugs.

Although this approach is still in its infancy, [Oren Rubin](https://twitter.com/shexman), CEO and founder at machine learning tool vendor Testim, says he believes that "the future holds a great opportunity to use this method to also automatically author tests or parts of a test. The value I see in that is not just about the reduction of time you spend on authoring the test; I think it's going to help you a lot in understanding which parts of your application should be tested."

ML does the heavy lifting, but ultimately a human tester does the verification.

### 5. Creating more reliable automated tests

How often do your tests fail due to developers making changes to your application, such as renaming a field ID? It happens to me all the time.

But tools can use machine learning to automatically adjust to these changes. This [makes tests more maintainable and reliable](https://content.microfocus.com/software-test-automation-tb/top-7-test-automation-mistakes%20?utm_source=techbeacon&utm_medium=techbeacon&utm_campaign=7014J000000dVA4QAM).

For example, current AI/ML testing tools can start learning about your application, understanding relationships between the parts of the document object model, and learning about changes throughout time.

Once such a tool starts learning and observing how the application changes, it can make decisions automatically at runtime as to what locators it should use to identify an element—all without you having to do anything.

And if your application keeps changing, it's no longer a problem because, with ML, the script can automatically adjust itself.

This was one of the main reasons [Dan Belcher](https://twitter.com/belcherdan), co-founder of testing tool company Mabl, and his team developed an ML testing algorithm. In my recent interview with him he said, "Although Selenium is the most broadly used framework, the challenge with it is that it's pretty rigidly tied to the specific elements on the front end.

"Because of this, script flakiness can often arise when you make what seems like a pretty innocent change to a UI," he explained. "Unfortunately, in most cases these changes cause the test to fail due to it being unable to find the elements it needs to interact with. So one of the things that we did at the very beginning of creating Mabl was to develop a much smarter way of referring to front-end elements in our test automation so that those types of changes don't actually break your tests."

**REFERENCES or EXAMPLES**

https://www.ecanarys.com/Blogs/ArticleID/170/Examples-of-Selenium-Webdriver-Scripts