USE CASE: EXPEDITING PERFORMANCE TESTING WITH AI-BASED OBJECT IDENTIFICATION

by Kevin Surace, Appvance

Key statement: Quality engineering should include functional, performance and security quality, and thus testing, as a standard set of tasks at each build. We often think of performance as load testing, but in fact performance can instead be real-world performance with real-world average loads. And in a highly competitive space, an extra second can mean lost sales or worse.

As native mobile apps have proliferated so has their usage in ecommerce. From searching for products to completing purchases, "m-commerce" is now over 70% of all e-commerce. As with all e-commerce, speed of the app and perceived responsiveness is critical to keeping a user focused on the task and getting to a purchase. How critical is performance to ecommerce? Here are four stats:

- The highest ecommerce conversion rates occur on pages with load times between 0-2 seconds. (Portent, 2019)
- The probability of bounce increases 32% as page load time goes from 1 second to 3 seconds. (Google, 2017)
- The highest ecommerce conversion rates occur on pages with load times between 0-2 seconds. (Portent, 2019)
- As page load time goes from one second to 10 seconds, the probability of a mobile site visitor bouncing increases 123%. (Google, 2017)

Continuous testing

If we want to be certain that our app performance is improving with every build, we must test at regular intervals. And with great statistical precision that ideally mimics the actual user experience. And these results would need to be automatically compared to prior builds as well as competitors apps. By performance we mean app performance and not load testing, which would likely be carried out at the API level rather than the UX. While we have been testing both performance and load of web

applications, doing so on native mobile apps is far more challenging. And in fact, the most common method used is literally a stopwatch.
While testers with stopwatches are a common method, the accuracy is poor and no better than +/- 1000ms. In order to improve an app at every turn we need to have accuracy at least 10X that of a human. And be able to test the same flows over and over again to average out network aberrations.

Our approach to providing native mobile performance testing

In AIQ we have a sophisticated AI based autonomous testing capability which was detailed in an earlier chapter. This unique technology can create its own tests (thousands) achieving very high application coverage with little to no human effort. But in the case of native mobile performance testing, we do not require the coverage of that engine. Instead, we know exactly what user flows we need and want to compare again and again.

It is important to not confuse performance testing with load testing. Performance is the step by step responsiveness of the app to the users eyes under normal load conditions. Load testing is to determine the scalability of an application, usability under large load, and perhaps ultimately where the app may fail under load. Both of these are critical in all app development. But faster performance to the user results in improved brand perception, improved productivity and increased sales in the case of ecommerce.

Here is where AI/ML, and specifically image or object recognition, comes into play. We will need to measure exactly what a human user will experience. And they experience it literally on the screen of a smartphone. Not Xpaths, not server responses. The screen itself.

This here is where a machine excels. Because we can utilize a variety of algorithms for image recognition to recognize aspects of a screen as a human would. For instance, a buy button or the like. If we want to know exactly the time between searching for an item to when it shows up to when there is a buy button to when the cart has the item in it (and so on) I must see it as the user sees it.

A human can recognize a BUY button for example regardless of where it is on the page. Even if it were to move around or be placed sideways. In essence we "clip" that object in our minds to locate it and execute it. So the process for a machine is the same. We need to train the machine to recognize an object taken out of context, and then look for that object once its placed back in context. In this feature set, we have utilized several techniques from CNN[1]'s to SIFT to SURF.. Starting with an actual "clip" of the target image, we use this to train our model. While CNN's need a fair but of data, SIFT and SURF can be essentially trained with one clip of an object. All of these run on GPU's at a rapid speed, which is required given the number of screens which must be reviewed by the machine.

[1] Convolutional Neural Networks

Applying native mobile performance testing at a large ecommerce player

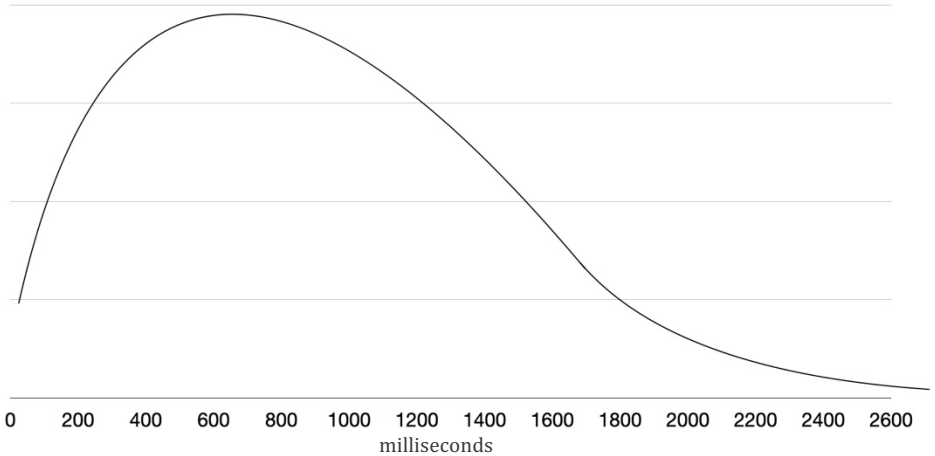
A large retailer has several Ecommerce apps which are native mobile. They compete in a highly competitive category around the world. How fast the app responds directly correlates to likeliness to complete a purchase. Measuring the timing of critical Ecommerce user flows, point to point, in their own mobile apps as well as competitors is critical in a hyper competitive space. Fast and accurate data can drive improvements with each build.

A new feature was developed leveraging the above image recognition algorithms leveraging cloud GPU's. The system recognizes screen elements in iOS and Android as a human would and reacts similarly to the way a human would. With the ability to operate any application and accurately time actions to available items on the next screen. Including cached and server requested items. The result is rapid accurate data that improves builds and increases sales.

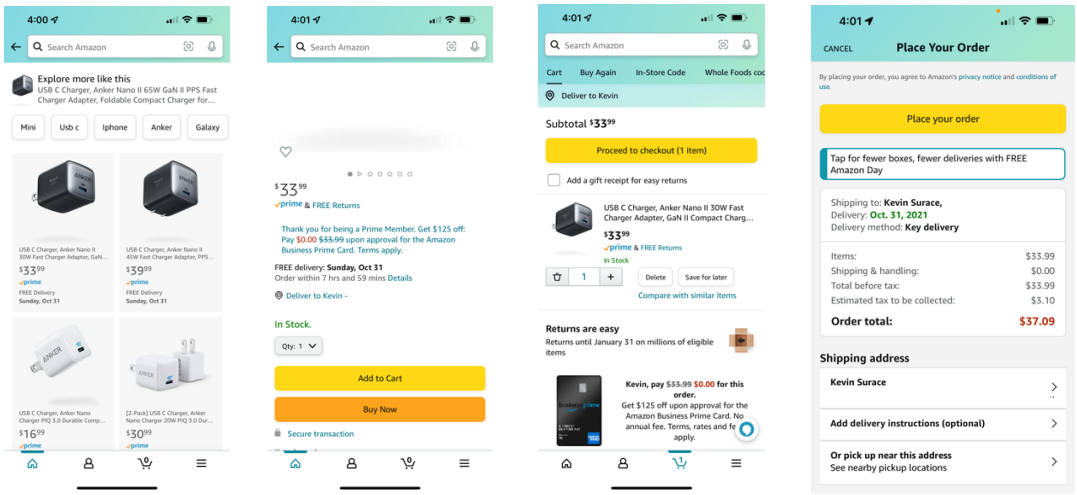
Specifically, we mimic the phone screen on a windows computer and measure the delays in the screen duplication as well as execution of steps. This data is used to adjust the measured timing.

The system records the mobile screen at 30 frames per second (or one snapshot every 33.33ms). Each frame contains encoded timing data as well as the screen replication itself. This is captured and transcoded as black and white since color information is not required in measuring performance timing.

A specific user flow with one app on a single device is run 1000X, from launching the app (which is measured) to nearly completing the purchase. Executing (automatically) the same user flow 1000 times generates an asymmetrical bell curve of results that frequently resembles this:



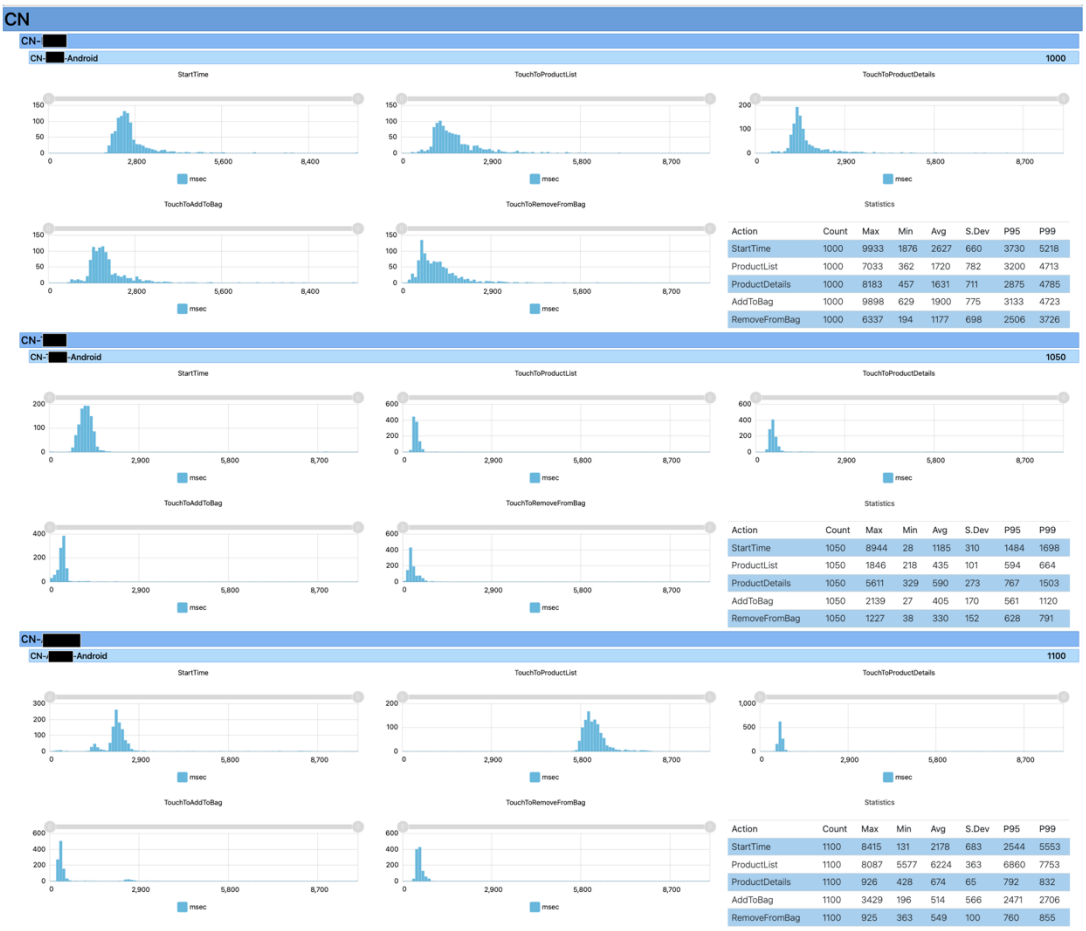
User Flows:



Above is an example of one user flow starting from launching the app to search to product to add to cart to proceed to checkout and finally placing the order. Many of these elements are embedded in the app code itself and can appear in a few milliseconds while other information may arrive from the server to properly form the next screen.

The image recognition is trained to look for specific screen elements which a user might interact with or use to determine that screen was ready to use. The image recognition processing is performed automatically as a batch rather than in-line with testing as dedicated cloud resources can be utilized post user flow runs. The number of screens to evaluate is large and can be calculated as 30 frames per second. If each step is an average of two seconds and there are 5 screens to evaluate that's 300 frames of interest in one user flow. As each flow is run 1000X that's 300,000 frames to look for image matches in for one device/app combination. This large number of data points allows for accuracy and statistical significance when evaluating the results. Multiply this by the number of apps across various countries and devices (in this case, the client requires 30 device/app combinations) and we end up evaluating ten million frames in any run. While this cannot be done quickly, we get absolutely accurate and comparable results. Comparable to prior builds and competitor apps.

After at least 1000 automated runs the QA team can retrieve highly detailed dashboards to compare performance between devices or competitors or builds. An example of a results dashboard looks like this:



In this case we can see comparisons of three different apps all with results from a single location in China.

Results

- 97% Improvement in timing accuracy
- 1000X The number of times a user flow re-runs to achieve statistical distribution
- 99% Improvement in statistical significance
- 100% Reduction in manual labor
- 33ms The timing accuracy of the image recognition technology
- 9 Number of apps evaluated
- 3 Number of countries with devices under test
- 10M+ Screens imaged processed and matched with AI

Summary

There are many reasons to want to understand the user experience on mobile apps, from the user's perspective. In m-commerce actual revenue is at stake. Milliseconds can make a difference. While we have had other UX driven approaches, none was designed to provide the curve of "to the eyes" results across 1000 runs. When we have this level of accuracy, we

can easily surface issues in any new build, and compare our results to those of the competition. This unique type of performance testing, relying on image recognition, allows us to improve our mobile app performance where it matters most…to the user.

About the author

Kevin is CTO and Chair of Appvance.ai, a leader in AI based autonomous testing. He has been awarded 93 worldwide patents.

https://en.wikipedia.org/wiki/Kevin_Surace
Twitter: https://twitter.com/kevinsurace
Linkedin: https://www.linkedin.com/in/ksurace/

About Appvance

Appvance.ai is the leader in AI-driven test generation, which is revolutionizing how software testing is performed. The company's premier product is Appvance IQ, the world's first AI-driven, unified test automation system. It helps enterprises improve the quality, performance and security of their applications, while transforming the efficiency and output of testing teams. Appvance.ai is headquartered in Santa Clara, California, with offices in Costa Rica and India.

Visit us at
https://www.linkedin.com/company/appvance/
https://twitter.com/appvance
https://www.facebook.com/AppvanceInc/