CSE 565 - Software Verification Validation and Testing Assignment 3

Kedar Sai Nadh Reddy Kanchi
School of Computing and Augmented Intelligence
Arizona State University, Tempe
kkanchi@asu.edu

WHAT IS DESIGN OF EXPERIMENTS (DOE)?

The term "design of experiments" (DOE) refers to a subfield of applied statistics dealing with the planning, execution, analysis, and interpretation of controlled experiments in order to determine the variables that affect the value of a parameter or set of parameters. DOE is a powerful and flexible data collection and analysis tool that can be applied to a wide range of experimental situations. It facilitates the manipulation of multiple input factors in order to determine how they affect a desired output (response). By adjusting several inputs at the same time, DOE can uncover significant interactions that would otherwise go unnoticed when experimenting with a single factor at a time. All possible combinations can be investigated (full factorial) or only a subset of the possible combinations can be investigated (fractional factorial).[6]

A well-planned and executed experiment can provide a wealth of information about the effect of one or more factors on a response variable. In many experiments, some variables are kept constant while the levels of another variable are changed. However, compared to altering factor levels simultaneously, this "one factor at a time" (OFAT) method of processing knowledge is ineffective. Blocking, randomization, and replication are essential ideas in designing an experiment.[6]

- **Blocking**: Blocking allows you to restrict randomization when randomizing a factor is impossible or too expensive by first running all trials with one setting of the factor before running all trials with the other setting.
- **Randomization**: It is easier to get rid of the effects of unidentified or uncontrolled variables when the sequence is random (the sequence in which an experiment's trials are run).
- **Replication**: Repetition of an entire experimental procedure, including the setup.

A well-performed experiment may provide answers to questions such as:

- What are the key factors in a process?
- At what settings would the process deliver acceptable performance?
- What are the key, main, and interaction effects in the process?
- What settings would bring about less variation in the output?

It is recommended to take a repetitive approach to learning, which typically involves the following steps:

- A screening strategy that minimizes the number of variables to be taken into consideration.
- A "full factorial" design that investigates how each combination of factors and factor levels affects the outcome in an effort to identify a set of values where the process is almost perfectly optimized.
- A response surface that is used to simulate the response.

WHEN TO USE DOE

When more than one input factor is suspected of influencing an output, use DOE. For instance, knowing how temperature and pressure affect a glue bond's strength may be desirable. Additionally, DOE can be used to confirm hypothesized input-output relationships and develop a predictive equation appropriate for what-if analysis.[6]

Experiences with DOE in Software Testing

DOE has been shown to achieve reasonable code coverage, so several companies have used it successfully in software testing.

Problem Statement

Based on the specifications for a new mobile application developed by the development team, the testing team is tasked with creating pairwise combination test cases using Design of Experiments. The testing team must conduct research and select a DOE-supporting tool before creating test cases for the provided specifications.

Pairwise Testing?

Pairwise Testing is a test design technique that ensures full test coverage. Pairwise testing is a type of black-box testing in which test cases are written to cover every possible combination of inputs. It is based on the observation that the majority of defects are caused by the interaction of two values. As a result, test coverage is improved by using input combinations.[3]

A software application's output is influenced by a variety of factors, including input parameters, state variables, and

environment configurations. Techniques such as boundary value analysis and equivalence partitioning can assist you in determining the possible values for individual factors. However, it is impractical to test all possible value combinations for each of those factors. As a result, a subset of combinations is generated in order to satisfy.[3]

Pairwise Testing is an extremely useful technique for designing tests for applications with multiple parameters. Tests are designed in such a way that for each pair of input parameters to a system, there are all possible discrete combinations of those parameters. The test suite does not cover all possible combinations, so it is not exhaustive, but it is extremely effective at detecting bugs.[3]

It is a subset of Combinatorial Testing (t-way testing). It is also known as All-pair testing.[5]

What is the objective of Pairwise testing?

- The primary goal of pairwise testing is to examine all possible pairs of test values to see if the application performs as expected.
- Create a subset of input/output values to create effective test coverage.
- To save time and effort in developing test cases that aid in bug detection.
- To create the best data set possible for the automation suite.
- To accelerate delivery while maintaining high quality.

When to use Pairwise testing?

According to a paper published on Combinatorial testing, experimental data shows that the interaction of two parameters causes nearly 60-95% of issues. Using Pairwise testing on 2-way combinations, we can detect a high percentage of errors[5]. We can use Pairwise testing:

- When there are numerous parameters, such as different input variables, several configurations are possible.
- When we are unable to conduct extensive testing due to unforeseen circumstances and the application is in a critical state, we can use this method.
- If the application has the potential to be automated.

What are the benefits of Pairwise testing?

- Pairwise testing can reduce testing time because we are working with a combination of inputs rather than a single one.
- It significantly reduces the number of test cases while still ensuring test coverage.
- It assists us in detecting bugs that we would have missed otherwise.
- It is concerned with maximizing impact while minimizing test case design efforts.
- There are several free and paid software tools available to help you generate value set pairs; you can do it without any manual intervention.

Limitations of Pairwise testing?

- It fails when the values selected for testing are incorrect.
- It fails when highly probable combinations get too little attention.
- It fails when interactions between the variables are not understood well.

Pairwise Testing Tools

Pairwise testing tools are simple test case generators that allow you to input and constrain a model of input parameters before generating test configurations based on the model. Because Pairwise testing is a complex procedure that can be time-consuming to perform manually on many input parameters, we use Pairwise testing tools. These tools are not only simple to use with a large number of input parameters, but they can also add constraints to the input parameters and generate test configurations as a result. To perform Pairwise testing, there are numerous tools available on the internet that applies the all-pairs testing technique that facilitates us to effectively automate the Test Case Design process by generating a compact set of parameter value choices as the desired Test Cases. Some well-known tools from the industry are:

- **PICT**: 'Pairwise Independent Combinatorial Testing', provided by Microsoft Corp.
- **IBM FoCuS** : 'Functional Coverage Unified Solution', provided by IBM.
- **ACTS**: Advanced Combinatorial Testing System', provided by NIST, an agency of the US Government.
- **Hexawise**
- **Jenny**:
- **Pairwise** : by Inductive AS
- **VPTag** : free All-Pair Testing Tool

Selected Tool for this Assignment

After going through the available tools as listed above, the tool that I have used for the assignment is PICT (Pairwise independent combinatorial testing). The PICT tool from Microsoft is a well-documented pairwise test generator that also supports many advanced features (Microsoft, 2019). PICT is an open source and cross platform program programmed in C++, and is single-threaded. The PICT tool is also compatible with a wide range of software systems, such as embedded devices, mobile apps, and web applications. It can be used as a versatile and adaptable testing tool for a variety of software development projects.[7]

Scope of the PICT tool

Despite being referred to as a pairwise test generator, PICT also supports higher interaction strength testing via a command line switch. PICT generates test cases and test configurations. PICT enables you to generate more effective tests than manually generated tests in a fraction of the time that hands-on test case design requires. PICT generates a subset

of test cases that include all pairwise input parameter combinations, testing for interactions between two input parameters simultaneously.

Purpose of the PICT tool

PICT generates a compact set of parameter value options that represent the test cases you should use to achieve complete combinatorial coverage of your parameters. The PICT tool also includes constraints, value weights, and other advanced features described in Czerwonka (2006). By reducing the number of test cases required to achieve thorough test coverage, PICT aims to assist testers in identifying flaws or vulnerabilities in software systems.

Example of the usage of the PICT tool

PICT is a command-line tool. The input model to PICT is a file that specifies the parameters of the SUT and their possible values. The figure below depicts an example of PICT's input model, as well as the use of constraints. This example is from the PICT documentation.[7]

```
#Comment

Type:           Single, Spanned, Striped, Mirror, RAID-5
Size:           10, 100, 1000, 10000, 40000
Format method:  Quick, Slow
File system:    FAT,FAT32,NTFS
Cluster size:   512,1024,2048,4096,8192,16384
Compression:    On,Off

#Constraints

IF [File system] = "FAT"   THEN [Size] <= 4096;
IF [File system] = "FAT32" THEN [Size] <= 32000;

IF [File system] <> "NTFS" OR
  ( [File system] =  "NTFS" AND [Cluster size] > 4096 )
THEN [Compression] = "Off";

IF NOT ( [File system] = "NTFS" OR
       ( [File system] = "NTFS" AND NOT [Cluster size] <= 4096 ))
THEN [Compression] = "Off";
```

Figure 1: PICT's input model.

Let us deconstruct the example in Figure 1 as a scenario in which we are creating a disk partition creation test suite; the domain can be described by the following parameters: Type, size, file system, format method, cluster size, and compression are all variables to consider. Each parameter has a limited number of possible values. For example, compression has only two options: On or Off; other parameters are made finite via equivalence partitioning (e.g. Size). This declaration of parameters and values is followed by a section detailing constraints. Constraints, as we can see, are conditions on parameter values used to ensure that no test is generated that violates these constraints.

For such a model, thousands of test cases can be generated. It would be impossible to put them all through their paces in a reasonable amount of time. Rather than attempting to cover all possible combinations, we decide to test all possible pairs of values. One pair, for example, is Single, FAT, and the other is 10 Slow. As a result, one test case can cover a large number of pairs. Testing all pairs is a more effective and less expensive alternative to exhaustive testing, according to research. It will provide excellent coverage while keeping the number of test cases to a minimum.

Working Example of the Pairwise independent combinatorial testing

Assume a Car Ordering Application with the following requirements[3]:

- The car ordering app allows you to buy and sell cars. It should support trading in Delhi and Mumbai.
- Registration numbers, whether valid or invalid, must be included in the application. There are around 5000 valid registration numbers. The following vehicles should be able to be traded in: BMW, Audi, and Mercedes.
- Both e-booking and in-store booking are available.
- Orders can only be placed during trading hours.

So, taking these requirements into consideration, we have can have the parameters as

- **Order Category**: Buy, Sell
- **Location**: Delhi, Mumabi
- **Car Brand**: BMW, Audi, Mercedes
- **Registration Numbers** : Valid, Invalid
- **Order type** : E-Booking, In-store
- **Order time** : Working hours, Non-working hours

So, if we want to test all the possible valid combinations, then it would be = 2 * 2 * 3 * 5000 * 2 * 2 = 240000 Valid test cases combinations. There is also an infinite number of invalid combinations.

Now if we simplify in the following terms :

- Use a well-chosen representative sample.
- Use groups and boundaries even if the data is not discrete.
- Finally, let us reduce the registration numbers to valid and invalid instead of considering 5000 valid registration numbers in our combinations.

Therefore now, if we want to test all the possible valid combinations, then it would be = 2 * 2 * 3 * 2 * 2 * 2 = 96 Valid test cases combinations.

We would need to test all possible combinations of options to thoroughly test the application, which would be extremely time-consuming and resource-intensive. Using Pairwise independent combinatorial testing, we can generate a subset of test cases that cover all pairwise combinations of options (PICT). PICT can generate the following test cases:

- Buy, Delhi, BMW, Valid, In-Store, Working Hours
- Sell, Mumbai, BMW, InValid, E-Booking, Non-Working Hours
- Buy, Delhi, BMW, Valid, In-Store, Non-Working Hours
- Buy, Mumbai, Audi, Valid, E-Booking, Working Hours
- Sell, Delhi, Audi, InValid, In-Store, Non-Working Hours
- Sell, Mumbai, Audi, InValid, E-Booking, Working Hours
- Buy, Delhi, Mercedes, InValid, E-Booking, Working Hours
- Sell, Mumbai, Mercedes, Valid, In-Store, Non-Working Hours

So, we have reduced the subset to 8 test cases from 96 test cases but still with complete covergae.

Kind of PICT tool used for the Assignment

There are two different kinds of PICT tools available.

- **Online Version**: Pairwise Pict Online - An online service that easily generates pair-wise test cases. It's powered by Microsoft Pict under the hood.
- **Offline Version**: PICT - PICT runs as a command line tool

For this assignment I have used the offline version. So for this, I have visited the PICT releases web-page and downloaded the executable (.exe) file and used this executable (.exe) file to generate pairwise combination test cases based upon the given requirements.

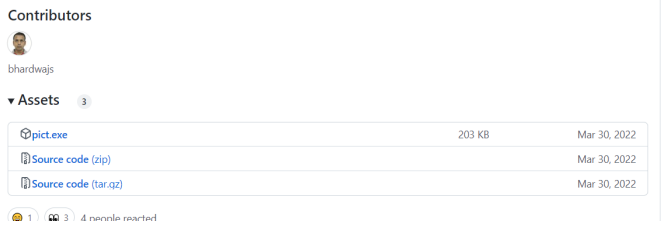

Figure 2: PICT offline latest version available releases

Creation of Test Cases

The given requirements for the newly developed mobile application are clearly mentioned in the figure below.

Type of Phone	Parallel Tasks Running	Connectivity	Memory	Battery Level
iPhone X	Yes	Wireless	1 GB	< 20%
iPhone 8	No	3G	2 GB	20 -39%
Samsung S9		4G	4 GB	40 - 59%
Huawei Mate		Edge	6 GB	60 - 79%
Google Pixel 3				80 - 100%

Figure 3: Given requirements for the mobile application

So, taking these requirements the first step that I did was to develop the model file that would act as the input to the PICT tool.

Creation of Input Model File

A model consists of the following sections: parameter definitions, [sub-model definitions], and the [constraint definitions] as you can see in the figure below[4]. Model sections must be specified in the aforementioned order and cannot overlap. Sub-models and constraints should come after all parameters have been defined. The section on sub-model definition as well as the constraints are optional. Sections do not need any extra separators. Lines with no content are allowed to appear anywhere. Comments are allowed, but must be followed by the # symbol. To make a basic model file, list the parameters one at a time, separated by commas: The comma is the default separator, but you can change it with the /d option.

```
parameter definitions
[sub-model definitions]
[constraint definitions]
```

Figure 4: Syntax to be followed for the input model file

Therefore, following the specified syntax, I have created my input model file (named CSE565_Assignment3_paramters) as a text file as you can see in the image below.

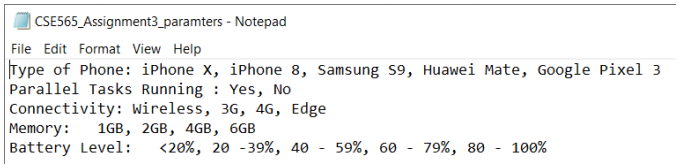

Figure 5: CSE565_Assignment3_paramters.txt file

Generating the test cases

Now with the input model file created, I have now opened my Command Prompt and navigated to the folder in which I have my executable (.exe) file. After navigating to the folder, I run the command ".pict CSE565_Assignment3_paramters.txt" and the test cases are generated as you can see in Figure 6.

Because of the readability issue of the output in the command line, I have saved the output in a different text file by running the command ".pict CSE565_Assignment3_paramters.txt ¿ GeneratedTestCases.txt" as you can see in Figure 6. And from the text file, the output has been moved into an excel file for better readability (which you can see at the end of this report) by running the command "./pict CSE565_Assignment3_paramters.txt ¿ GeneratedTestCases.xls" as you can see in Figure 6 .

So, to summarize, the number of test cases Generated by DOE tool i.e. the PICT tool are 25.

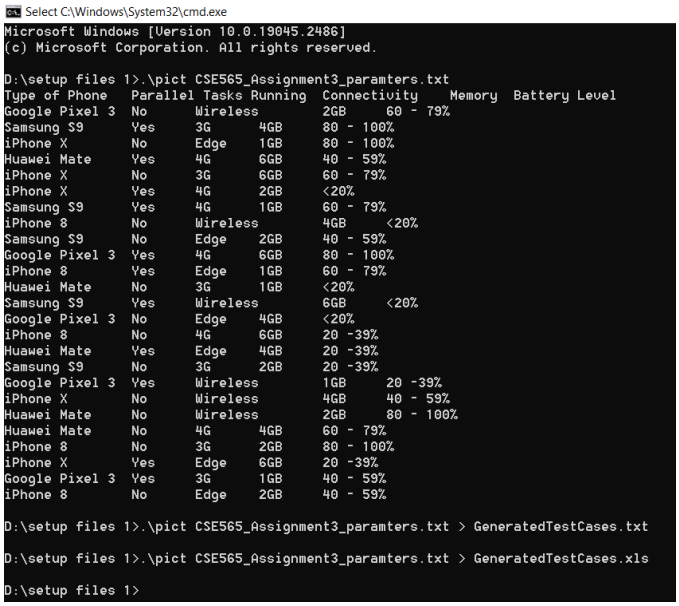

Figure 6: Generated Test Cases in Command Line Interface

Assessment of the PICT tool

Usability of the PICT tool

The PICT tool is extremely useful because it can help software testers create a comprehensive set of test cases that cover all possible input combinations while reducing the number of test cases needed. The following are some examples of how PICT usability can benefit software testers[4]:

- **Saves time and effort**: PICT can help reduce the number of test cases required to test a software system by identifying only the most important combinations of inputs that could affect the output. By testing only the most important combinations based on the user's inputs and constraints, PICT can reduce testing time and effort. PICT can generate test cases that only cover the most important input combinations, such as AB, AC, AD, BC, BD, and CD, if a system has four inputs (A, B, C, and D). By testing these combinations, PICT can reduce the number of test cases required to test the system.
- **Comprehensive test coverage**: PICT can help improve test coverage by identifying all possible input combinations that could affect a system's output. By testing all possible combinations, PICT can ensure that all interactions between inputs are covered while minimizing the number of test cases required. For example, if a system has four inputs (A, B, C, and D), PICT can generate test cases that cover all possible input pair combinations, such as AB, AC, AD, BC, BD, and CD. By testing these combinations, PICT can enhance the system test coverage.
- **Flexibility**: PICT can be used in a wide range of industries, including software, hardware, and manufacturing. It can help identify input combinations that may affect a system's output, regardless of domain. PICT, for example, can be used in manufacturing to identify combinations of input factors such as temperature, pressure, and time that may affect the output of a manufacturing process. Another example can be to automate testing and generate reports, PICT, for example, can be integrated with Visual Studio.

Coverage of the PICT tool

The PICT (Pairwise Independent Combinatorial Testing) tool generates test cases by testing all possible input parameter combinations for a given system or application in a systematic manner. The tool uses a combinatorial testing approach to ensure that all possible combinations of input values are tested with the fewest number of test cases. The ability of the PICT tool to generate test cases that cover all possible combinations of input parameters is referred to as coverage[4].

PICT coverage is determined by pairwise combinations of input parameters. Pairwise combinations are created by selecting two parameters at a time and generating all possible value combinations for those parameters. This method can help detect defects caused by interactions between pairs of input parameters, which are common in complex software systems.

The coverage of PICT can be explained with the following example: Assume we're testing a system with three input parameters: A, B, and C. Each parameter can have one of two values: 0 or 1. As shown below, PICT generates test cases that cover all pairwise combinations of the input parameters:

- **Test Case 1**: A=0, B=0, C=0
- **Test Case 2**: A=0, B=1, C=1
- **Test Case 3**: A=1, B=0, C=1
- **Test Case 4**: A=1, B=1, C=0

Each test case is intended to cover a unique pair of input parameters, ensuring that all pairwise combinations are tested at least once. This ensures that the interactions between the input parameters are adequately addressed.

Features of the PICT tool

PICT (Pairwise Independent Combinatorial Tool) is a software application testing tool. PICT generates test cases by analyzing possible input value combinations and using a mathematical algorithm to ensure that each pair of input values is tested at least once. PICT also has a number of advanced features that can help testers create more effective test cases[4].

- **Re-using Parameter Definitions**: You can reuse parameter definitions across models with this feature. For example, if you have a model for testing a web application's login feature and want to create another model to test the registration feature, you can reuse the username and password parameter definitions in both models. This can save time while also reducing errors during testing.
- **Sub-Models**: Sub-models can be created in PICT within a larger model. Sub-models can be used to represent different modules of software under testing. For example, if you are testing a large web application, you could create

a sub-model for each module, such as login, registration, and profile. This enables you to test each module independently while also thoroughly testing the interactions between modules.

- **Aliasing**: Aliasing is a feature that combines parameters and tests them as if they were a single input. This is useful when testing parameters that are highly dependent on one another, such as the date and time of an event. By aliasing these parameters, you can test them as if they were a single input, reducing the number of test cases required.
- **Negative Testing**: The process of looking for errors or unusual inputs is known as negative testing. In PICT, you can specify negative tests to help you find errors or unexpected behavior in the software application. If you're testing a form that requires numeric input, for example, you can include a negative test that enters a non-numeric value to see how the application reacts.
- **Weighting**: Weighting is a feature that allows you to assign varying degrees of importance to different parameters or parameter combinations. This can be useful when testing the software application for more critical scenarios. For example, if you're testing a payment system, you might want to give more weight to the combination of parameters that represents a large payment amount.
- **Seeding**: Seeding is the process of specifying a starting point for the PICT algorithm. When you want to ensure that specific parameter combinations are included in the test cases, this is useful. If you're testing a web application with different user roles, for example, you might want to seed the algorithm with a set of parameters representing an administrator. This can help ensure that the test cases cover all of the different user roles.

Assessment of the PICT tool in my own words

- PICT is designed to be simple to use for software testers with little or no test design experience.The PICT tool is straightforward to use and does not require extensive knowledge of combinatorial testing or coding skills. It is a command-line tool that requires very little input from the user.
- PICT supports multiple input formats, including CSV, Excel, and XML. This makes it easy to import test data from other sources, such as spreadsheets or databases.
- It can handle a large number of input parameters and constraints while reducing test case redundancy. Users can enter their testing requirements into the tool, such as the number of parameters, possible values, and constraints, and it will generate a set of test cases based on pairwise combinations.
- Testers can use PICT independently or integrate it into their testing framework via APIs. PICT also supports a number of programming languages, including C#, Java, Python, and Ruby.
- By testing all possible input parameter combinations, PICT provides comprehensive coverage. The tool is capable of testing a wide range of parameters, including Boolean, integer, and string variables. Pairwise testing reduces the number of test cases required to achieve the same level of coverage as testing each combination individually. As a result, it is an effective and useful tool for testing software applications.
- In addition to pairwise testing, PICT also supports higher order combinations such as triplets or quadruplets, which can provide even more comprehensive coverage.
- PICT's ability to handle constraints is a useful feature. Testers can define input parameter constraints to ensure that certain combinations of values are not tested. This comes in handy when testing software that has specific requirements or constraints.
- PICT allows users to specify factors and levels for input parameters, allowing them to create more efficient test cases. Furthermore, PICT can automatically detect redundant test cases, saving time and effort during testing.
- PICT enables users to customize the output format of generated test cases to meet their specific testing requirements. Users can thus create test cases that are compatible with their existing testing tools and processes.
- : Since PICT provides a command-line interface that allows users to create and execute test cases from a terminal window it makes it easier for the PICT tool to be integrated with other testing tools and frameworks such as Visual Studio, Selenium, and Cucumber to automate the testing process and generate reports.

Conclusion

To summarize, PICT is a powerful testing tool that can assist testers in creating test cases optimized for pairwise testing, covering a wide range of input parameters, and dealing with constraints. Its usability, coverage, and features make it a valuable tool for software testing, as it provides a variety of features that can improve the efficiency, effectiveness, and flexibility of the testing process, especially for projects with a large number of input parameters. Software developers and testers can use PICT to ensure that their applications are thoroughly tested and defect-free.

References

- Prof. James Collofello, CSE 565 Lecture Videos, 2023
- Prof. Ayca Tuzmen, CSE 565 Lectures, 2023
- Pairwise Testing Or All-Pairs Testing Tutorial With Tools And Examples
- Pairwise Independent Combinatorial Testing
- Pairwise Testing Guide: How To Perform Pairwise Testing
- WHAT IS DESIGN OF EXPERIMENTS (DOE)?
- EXTENDED COMBINATORIAL TESTING USING GRAPH ALGORITHMS AND APACHE SPARK

Type of Phone	Parallel Tasks Running	Connectivity	Memory	Battery Level
Google Pixel 3	No	Wireless	2GB	60 - 79%
Samsung S9	Yes	3G	4GB	80 - 100%
iPhone X	No	Edge	1GB	80 - 100%
Huawei Mate	Yes	4G	6GB	40 - 59%
iPhone X	No	3G	6GB	60 - 79%
iPhone X	Yes	4G	2GB	<20%
Samsung S9	Yes	4G	1GB	60 - 79%
iPhone 8	No	Wireless	4GB	<20%
Samsung S9	No	Edge	2GB	40 - 59%
Google Pixel 3	Yes	4G	6GB	80 - 100%
iPhone 8	Yes	Edge	1GB	60 - 79%
Huawei Mate	No	3G	1GB	<20%
Samsung S9	Yes	Wireless	6GB	<20%
Google Pixel 3	No	Edge	4GB	<20%
iPhone 8	No	4G	6GB	20 -39%
Huawei Mate	Yes	Edge	4GB	20 -39%
Samsung S9	No	3G	2GB	20 -39%
Google Pixel 3	Yes	Wireless	1GB	20 -39%
iPhone X	No	Wireless	4GB	40 - 59%
Huawei Mate	No	Wireless	2GB	80 - 100%
Huawei Mate	No	4G	4GB	60 - 79%
iPhone 8	No	3G	2GB	80 - 100%
iPhone X	Yes	Edge	6GB	20 -39%
Google Pixel 3	Yes	3G	1GB	40 - 59%
iPhone 8	No	Edge	2GB	40 - 59%