January 2020

# MACHINE LEARNING ASSISTED QUICK EYESCAN (MLAQE) FOR SIGNAL INTEGRITY CHECK

Jim Xu

Sreenivasa Rao Bandlamudi

Ankit Sood

Vinit Bansal

Ravi Eda

*See next page for additional authors*

## Inventor(s)

Jim Xu, Sreenivasa Rao Bandlamudi, Ankit Sood, Vinit Bansal, Ravi Eda, and Arun Singh

# MACHINE LEARNING ASSISTED QUICK EYESCAN (MLAQE) FOR SIGNAL INTEGRITY CHECK

AUTHORS:
Jim Xu
Sreenivasa Rao Bandlamudi
Ankit Sood
Vinit Bansal
Ravi Eda
Arun Singh

## ABSTRACT

Presented herein is a quick scan method for signal integrity checks through sampling of four points under certain thermal condition. The techniques presented herein use a machine learning trained model to ensure signal integrity check fidelity.

## DETAILED DESCRIPTION

Eyescan is a technology that is widely used today to debug complex Signal Integrity (SI) issues. Eyescan can provide valuable insight into the nature of signal imperfections that can lead to errors while a receiver attempts to interpret the value of an incoming bit. The basic eye measurement method takes samples around the center of a decision point in order to find the eye shape through Bit-Error-Rate(BER) calculation. Given the large number of sample points, the process of determining an acceptable eye window normally takes at least 25 minutes to finish, per lane, in software implementations which restrict its use for time-sensitive applications. Even with the possible implementation as firmware for faster speeds, this process will take certain amount of firmware space and also increases the firmware design complexity. In other words, traditional eyescan methods to determine the acceptable eye window take too long time (if implemented in software) and/or increases the size and complexity (if implemented in firmware). Such limitations limit the use of eyescan for time sensitive applications in real-time systems.

As such, presented herein are Machine Learning Assisted Quick Eyescan (MLAQE**)** techniques (Quick Eyescan techniques) that are designed to significantly reduce the complexity needed to determine an acceptable eye window. The MLAQE techniques presented herein:

(1) Use typical samples along with the X axis and Y axis of a center eye, instead of full scale samplings to cut the number of sampling points and to reduce calculation time.

(2) Use a machine learning algorithm to train the eyescan model to ensure eye measurement reliability with less sampling points as described in (1). In addition to modeling sampling points, the thermal conditions are also part of the modeling.

(3) Apply the trained eyescan model for eye window calculation to achieve reliable eye window measurement with much less sampling points and much shorter calculation time to meet real-time system requirement

The MLAQE techniques presented herein reduce the software calculation time for determination of the eye window from multiple minutes to less than a second, without losing fidelity. As a result, the MLAQE techniques presented herein may be used for time sensitive applications, such as faster link bringup and recovery. In addition to these software implementation advantages, the same trained quick eyescan model can be used in a firmware design with less complexity and less space usage.

The Figure 1, below, demonstrates the traditional full scale eyescan scheme, as well as the MLAQE techniques presented herein.

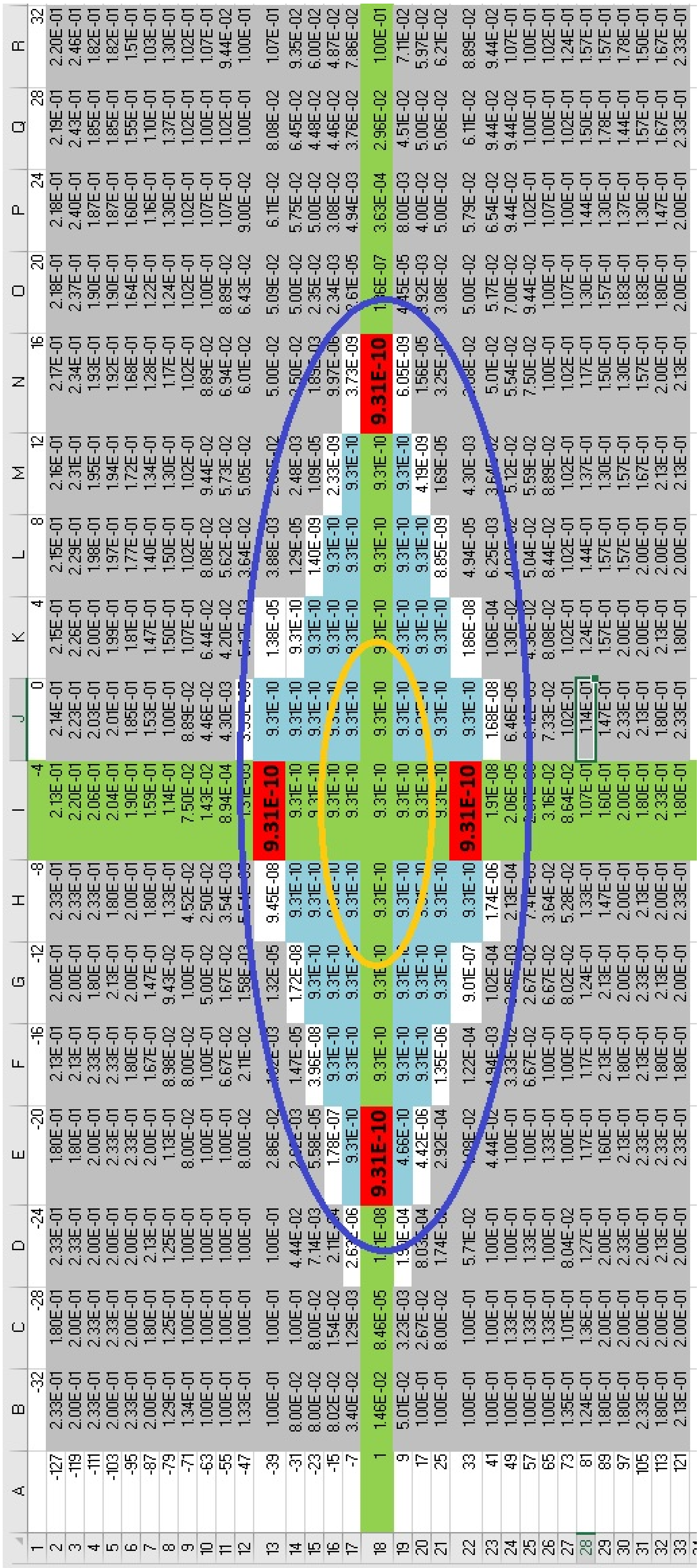

Figure 1

In the traditional full eyescan, from the center point (row 18, col I) in Figure 1, sampling points are selected with step size 4 in the X axis and step size 8 in the Y axis to measure the Bit Error Ratios (BER). The BER of $e^{-10}$ is set as the threshold so that all sampling points within the threshold BER of $e^{-1}$ form a good eye window (marked in light blue color). The total 32 x 17 points are sampled to calculate the BER in order to accurately form the good eye window. This may take more than 25-30 minutes to complete the calculation for a 25G lane. The step size selection also affects the calculation (i.e., the less step size, the more sampling points, thus the more accurate eye window but the longer calculation cycle).

In contrast to the traditional full scale eyescan, the MLAQE (Quick Eyescan) techniques presented herein operate to pick up the typical sampling points along the X axis and Y axis only, starting from the center point. For the example, as shown below in Figure 1, the Quick Eyescan only picks up the sample points from row 18 (X axis from the center point) and column I (Y axis from the center point). These Quick Eyescan points are marked as the green background in Figure 1. This reduces the total sampling points from “32 x 17” to “32+17.” As such, the calculation time is reduced to less than a second.

In Figure 1, the four points marked in bold (with red background) in the X axis and Y axis form an eye window which almost overlaps the eye window formed by a traditional full scale eye scan. The four points are determined based on the BER threshold crossing of $e^{-10}$.

In addition, the MLAQE techniques presented herein define a Passing Eye window (as shown by the ovals in Figure 1). If the Passing Eye window is determined to coincide with the golden-colored oval in Figure 1 (which is contained by the measured eye window), then good signal integrity is present. However, if the Passing Eye window is determined to coincide with the navy-colored oval in Figure 1 (which is not contained by the measured eye window), then good signal integrity is not present (i.e., this indicates poor signal integrity).

As noted, the MLAQE techniques presented herein help reduce the calculation time for determination of the eye window. However, reduction in sampling points may lead to a smaller eyescan window that makes it more difficult to contain the Passing eye window.

This could lead to a false determination of poor signal integrity. The techniques presented herein address this issue through the use of a trained machine learning model.

In general, the MLAQE techniques obtain four sampling points (two on X axis, and two on Y axis), where their neighbor sample points in the direction away from the center point cross the BER threshold. Instead of using these points to form an eye window for comparison with the Passing Eye window, these four sample points are used as input features to a Linear regression model (or a 3-layer Neural Network model) to predict acceptable signal integrity.

As shown in the Figure 2, below, the columns B/C/D/E define the four features which map to the four sampling points where their BERs are about to cross. Their values are recorded to be the distance to the center eye. The first sample (at row one) is the direct mapping of data collected/calculated in Figure 1.

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Training samples | Feature-1: X-axis left x1 | Feature-2: X-axis right x2 | Feature-3: Y-axis top x3 | Feature-4: Y-axis bottom x4 | Center Eye | Temperature x5 | Signal Integrity |
| 2 | 1 | -20 | 16 | -39 | 22 | (-4, 1) | 35C | Passed (1) |
| 3 | 2 | -24 | 12 | -31 | 33 | (0,1) | 36C | Passed (1) |
| 4 | 3 | -24 | 16 | -39 | 41 | (0,1) | 34C | Passed (1) |
| 5 | 4 | -20 | 20 | -31 | 33 | (0,1) | 32C | Passed (1) |
| 6 | 5 | -10 | 10 | -20 | 22 | (0,1) | 36C | Failed (0) |

**Figure 2**

The eye window measurement may be affected by thermal conditions. As such, the techniques presented herein may add the component temperature as the fifth feature, as shown in column G of Figure 2, to be taken as an input to the machine learning model. Column H records whether the result is Passed (1) or Failed (0), based on the measurement. Column F records the center eye coordinates, for the adjustment purpose of the four features values (column B/C/D/E), but not directly used for features as input to the Machine learning model.

The same measurements as in Figure 1 may be repeated to find the four optimal sample points and the component temperature. The result is recorded in the format of Figure 2. The techniques presented here may either reload the system or inject errors on purpose so that it can cover both PASSED and FAILED scenarios in the repeated experiments. The same fault injection testing can be repeated in the environmental chambers so as to obtain the result under different thermal conditions. The output will be a table similar Figure 2 with four feature values plus temperature reading and a corresponding Passed/Failed output. Each experiment result is shown as a row which forms the training data to find the optimal weights for each feature thru the machine learning algorithm.

In certain examples, the Logical Regression may be used to build the training models. Assuming four features as *x1, x2, x3, x4, x5*, corresponding to row B to row E, and row G, it is possible to have *m* rows in the table corresponding to the total number of *m* experiments. By applying the logical Regression algorithm, it is possible to find the optimal weights array *[W0, W1, W2, W3, W4, W5]* (*W0* is the bias) so that the system can use (x) to predict Passed (1)/Failed (0) where:

$$h_w(x) = g(W0 + W1x1 + W2x2 + W3x3 + W4x4 + W5x5)$$

and

$$g(z) = 1/(1 + e^{-z})$$

In summary, the MLAQE techniques presented herein:

- select four optimal sample points under various thermal conditions through our quick scan method.

- Train the Eyescan model by using the four selected points and measured temperature as features for machine learning algorithm to obtain the optimal weights array for each feature.
- Use the trained Eyescan model in online calculation only for simplicity and accuracy.

With logical regression functions, the trained model can achieve 95% or above accuracy. The prediction accuracy can be further improved to 99% if non-linear neural networks are used instead. It is noted that the training is performed to find the optimal weight array, which imposes no performance impact on running time for a real-time system.

The trained weights array [W0, W1, W2, W3, W4. W5] are directly programmed into software/firmware for Passed/Failed prediction. During run-time, the software/firmware uses the quick eyescan scheme to first find the four typical sampling points (X1, X2, X3, X4) and the component temperature X5 at measurement time, then use following formula to compute:

$h_w(x) = g(W0 + W1X1 + W2X2 + W3X3 + W4X4 + W5X5)$

If $h_w(x) > 0.5$ *meaning a Passed, Otherwise a Failed.*

The g(z) calculation may be complicated for some real-time system/firmware but the g(z) curve basically shows that g(z) is almost 1 for z > 4 and is almost 0 for z < -4. In other words, we only need to calculate:

$z = W0 + W1X1 + W2X2 + W3X3 + W4X4 + W5X5$
in the runtime system.

This helps simplify the calculation so that for z outside the window [-4, 4], the software/firmware can make a sure prediction, and for z within [-4,4] window, use the training data cut a line for Passed/Failed classification without going to the real g(z) calculation. The trained weights may be updated based on further sampling in the real time testing or field execution. The revised weights can be put into use through software/firmware upgrade. Or simply make them programmable as registers in the case of firmware.

If the running system has enough computing power, the machine learning model training can be put online as well (e.g., each time a link up scenario provides a PASSED sample, or a link down scenario provides an FAILED sample). In either case, the system samples the four typical eyescan points and temperature to build up the machine learning sample database on the fly. Each time the database is updated, the system runs the machine learning algorithm to rebuild the model and reprogram the rebuilt model into the software/firmware. The online model training helps build per-board based model for SI check at the cost of CPU and disk resources. This will further improve the SI check accuracy, given the model training/online model use assumes that all boards with the same type show the same SI characteristics.

It is a common practice to set the chip-to-chip drive strength for signal integrity in board design. These drive strength parameters are calculated and measured in the lab and provided to software to program them into the chip to ensure chip-to-chip signal integrity. This method has fundamental difference from the MLAQE proposed here, namely:

- The drive strength parameters are static values to be programmed into the component's registers to ensure signal strength between chips, mostly from the sender side. In both sender side and receiver side SI programming cases, they provide no way ensuring good receiving signals. These parameters are best-effort calculated values from the sender side only. They are statically used in the field.
- The MLAQE helps detect signal integrity on the receiving side. It does not directly use the calculation result from the lab, but instead builds an eyescan model through machine learning. The trained model is programmed, but it is combined with real-time dynamic eyescan measurement results and thermal conditions to provide much more accurate detection for signal integrity.
- The MLAQE can also help improve robustness of traditional static SI drive strength setting. Instead of providing a group of static SI parameters for software to programming, the SI hardware team can provide a machine learning trained model, and the software will apply the model to the real time scenario to decide what SI

parameters should be programmed into the chip. This will ensure more robust SI settings.

Figure 3, below, illustrates an offline model training with online use, in accordance the techniques presented herein.

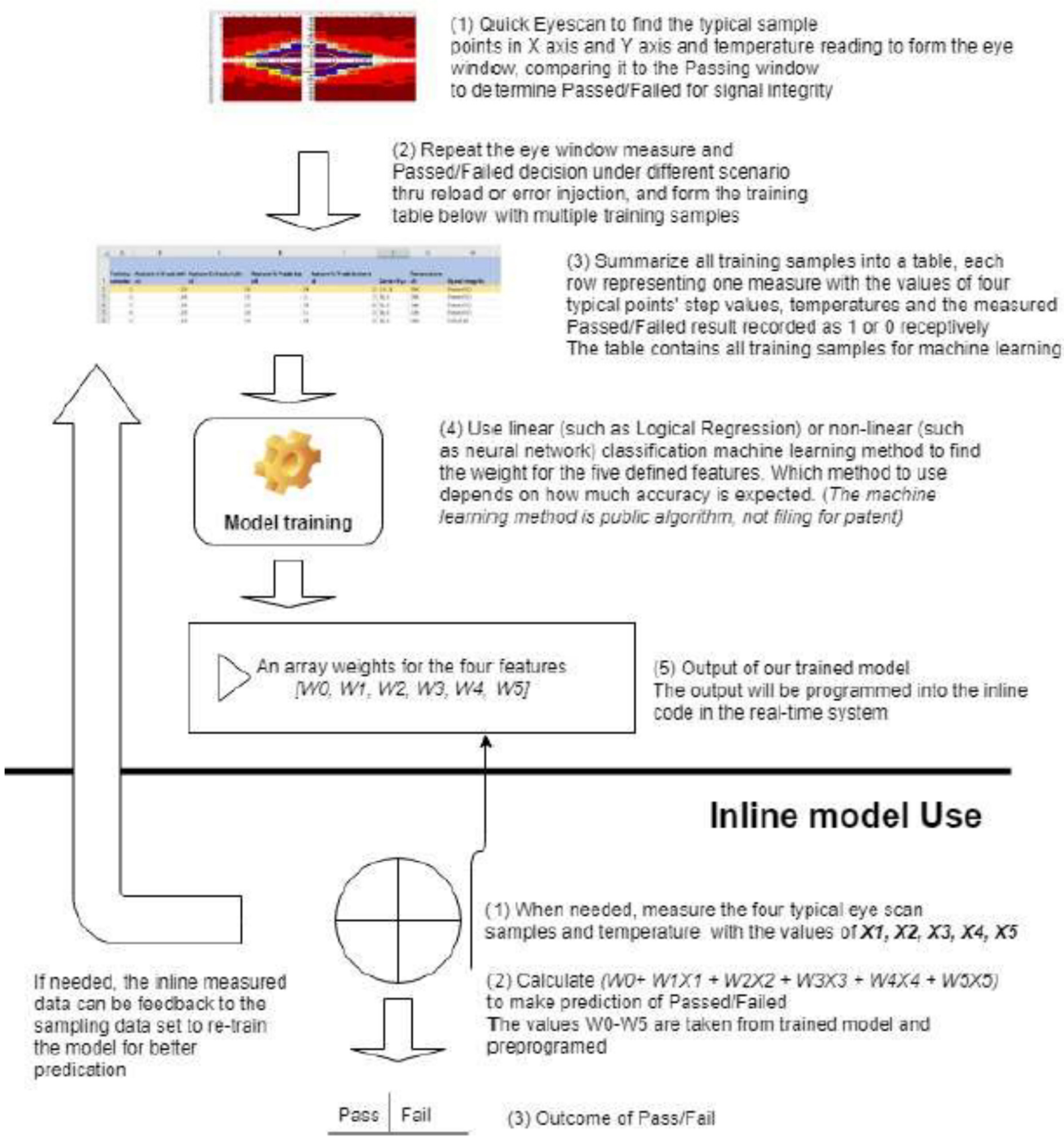


**Figure 3**

Figure 4, below, illustrates online model training with online use, in accordance the techniques presented herein.

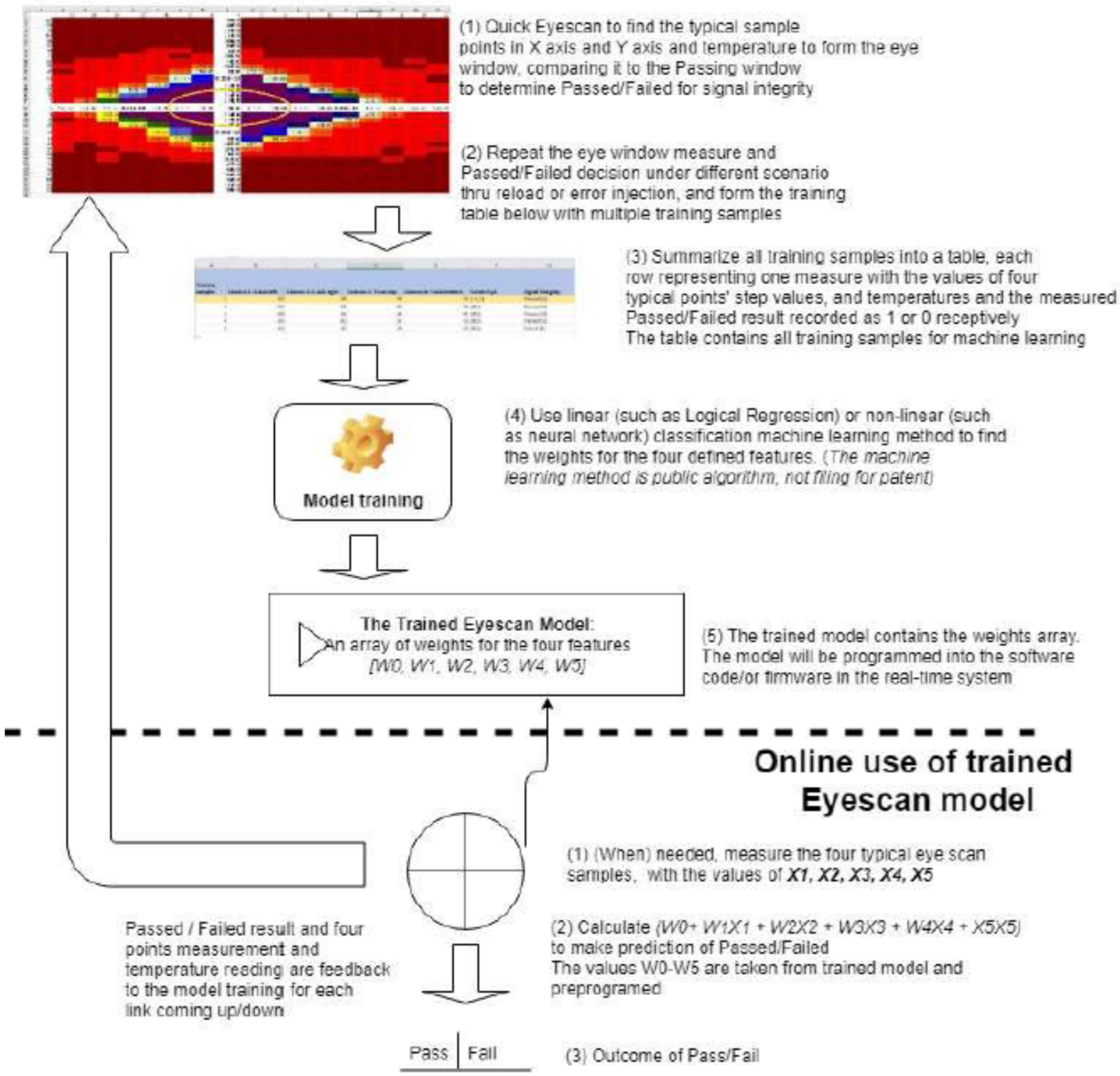


**Figure 4**