ISyE 6740 – Spring 2023
Final Project

Team Member Names: Daniel Hardiman

Project Title: Image Classification for Wafer Defect Detection

Problem Statement:

Defect detection is a problem of critical importance in semiconductor manufacturing. When building devices on the nanometer scale, there are many different things that can cause an individual die or an entire wafer to be compromised which are entirely imperceptible to humans. Large amounts of resources in the manufacturing process are dedicated to measuring and inspecting the wafers at different steps along the way in order to ensure that any adverse impacts to the wafers can be detected quickly. The importance of this scales accordingly with the volume of wafers that a factory is producing: a higher rate of wafers running through the line means more wafers exposed to a defect-causing mechanism before the issue can be detected and contained. Furthermore, many of the types of inspection tools that are commonly used for this purpose have capacity limitations that don't allow for anywhere near 100% of wafers to be scanned. While there are sampling strategies in place to mitigate this, it's always possible for intermittent issues to go undetected.

Some inspection tools/modules in the manufacturing process, however, do have the capability for 100% sampling. One of these module types which is used at Global Foundries Fab 8 is called the wafer intelligent scanner (WIS). The WIS is an inline module that is part of the lithography tools – these tools spin-coat the wafers with photoresist, transfer the pattern to the wafers via exposure in the stepper, develop away the non-crosslinked resist, and take a picture of the full wafers in the WIS before sending them on to be etched or implanted. These images can be used to detect various macro-scale defect signatures. The images collected by the WIS are analyzed in real time using inspection recipes that are chosen based on certain context associated with the lot. While these recipes can be effective at picking up true defect signatures, they have a few inherent limitations: 1) Known bad wafers are needed to be able to tune the recipes to catch a particular signature, 2) some signatures are difficult to capture, even after tuning the recipes based on known bad wafers, and 3) due to some equipment software limitations, it isn't always possible to interlock a tool based on repeated fails for a given signature. Therefore, it isn't terribly uncommon to have cases where a large number of wafers are exposed to some defect-causing mechanism with a signature that is visible on the WIS images. The task then becomes to take this exposure of wafers (often numbering in the thousands to tens of thousands) and use the WIS images to determine which wafers are impacted with the signature. Several different pre-processing methods and classification algorithms will be tested with an existing dataset to determine the most effective strategy for identifying bad wafers.

Data Source:

In one recent event, over 8000 wafers were exposed to an intermittent dispense drip on one of the spin-coating modules of a particular tool before the drip was finally detected and contained. Out of these 8000+ wafers, 16 of them were impacted with a signature that was visible on WIS which showed up as poor coverage near the edge of the wafer (some had this around the full circumference, others just had it in a single quadrant). The images captured at the nearest litho process step with 100% WIS sampling after the step at which the drip happened will be used as the dataset for this analysis. The raw images themselves are too large to store in memory for all 8k+ wafers and too large to be used for training classification algorithms (2048x2048x3). Therefore, each image will be queried from the local server using a list of lot/wafer IDs, blocked-reduced, gray-scaled, and dimension-reduced via PCA to prepare for training/testing. Additionally, each wafer has a binary label indicating whether the wafer has a signature caused by the drip (1 = signature, 0 = no signature) which will be used for training the various classifiers.

The proposal for this report originally mentioned that multiple datasets from different tool events with different types of signatures would be tested. However, after starting to work with the data and test the different pre-processing methods, classifiers, and training strategies, it seemed like a better use of time to deep-dive on a single dataset as a proof of concept rather than try to stretch to cover more than one. Furthermore, this particular dataset was a good one to use for this purpose since the defect signature proved to be very hard to detect using standard inline measurements, so a classification algorithm which could reliably pick out impacted wafers from an unseen dataset via the WIS images would be highly valuable.

Due to confidentiality restrictions, the full data set will not be submitted with the final report, but a few example images will be provided to show what the drip signature looks like and the differences between the images in raw vs. pre-processed form.

Methodology:

The first test that was done looked at the raw images with no lot-based correction. This essentially served as a control test since the drip signatures are very hard to see on the raw images without subtracting the mean image for each lot, so the expectation would be that the classifiers should not be able to reliably distinguish the impacted wafers from the good wafers. Part of the reason for this test was also practical: to determine the amount of dimensionality reduction that needed to be applied to the images in order for all 8k+ to be stored in memory and for the model training to not take an excessively long time to run. The images ended up needing to be blocked reduced by a factor of 8 on each axis in order to be storable, and the dimensionality was reduced further by taking only the first 22 principal components (corresponding to a cumulative variance of ~95%). The gray-scaling in this test (and all other tests) was done by simply taking the mean of the RGB value, and then standardizing all the features by dividing the pixel values by 255 as per best practices prior to applying PCA. These images were then used to train neural network models with a range of different values for the L2 regularization constant (alpha) and the hidden layer sizes. None of the models tested were able to pick out the bad wafers, which was not surprising since the drip signatures were nearly impossible to discern on the raw images to the human eye, and much more of the variance between images was driven by the inherent differences between product groups than by the defect signatures. All of the images in this dataset happened to be from the same layer, but it is common for the exposures in these types of WIS bounding events to include multiple different layers, which would also naturally increase the variance between images for reasons unrelated to the defect signature intended to be captured.

	Predicted: 0 (Good)	Predicted: 1 (Bad)
Actual: 0 (Good)	8730	0
Actual: 1 (Bad)	16	0

Table 1. Confusion matrix for MLP classifier (alpha = 1000, hidden layer sizes = (40,)), trained and tested on full dataset (raw images).

The next test used the same degree of block reduction with the images, but instead of taking the raw values for each image, the lot reference image (i.e. take the arithmetic mean of the RGB images for all 25 wafers in the lot) was subtracted from each individual image before gray-scaling, standardizing the features, and performing PCA. Interestingly, the number of principal components that needed to be retained in this case to reach a cumulative variance of ~95% was quite a bit larger that of the raw images (80 vs. 22, respectively). This method of correction by subtracting the reference image is a heuristic which is very commonly used for these types of analyses as part of the short-term containment for WIS bounding events. More often than not, any signature that is visible on WIS but is difficult to see on the raw images becomes much more obvious when the lot reference image is subtracted, making the subsequent analysis to separate the bad wafers from the good much easier. One vulnerability of the lot average correction method is that it does not work well for signatures with a high hit rate at the lot level. For example, say if 12 or all 25 wafers out of a given lot had the signature, then taking the average of that lot and subtracting it likely wouldn't be helpful for making the signature more visible. This is the rationale for using reference images at the PG/layer level instead: calculate and store average images across X number of randomly selected recent lots for all the different combinations of product group and layer that are currently in production and use those average images for the correction. In theory, even if one or more full lots captured in the average image were impacted with a defect signature, the overall contribution would be small relative to the total number of lots used to create the image causing the signature to fade out. The PG/layer method was intended to be used for this project, but the script ended up running extremely slowly and the reference images that were produced looked much different than the lot-level reference images. Furthermore, since this drip case was intermittent, each impacted wafer was the only one in its respective lot, and therefore the lot-level correction was universally effective at making the signatures visible. See Figure 1 below for an example of the corrected and uncorrected versions of the same image side by side from one of the bad wafers.

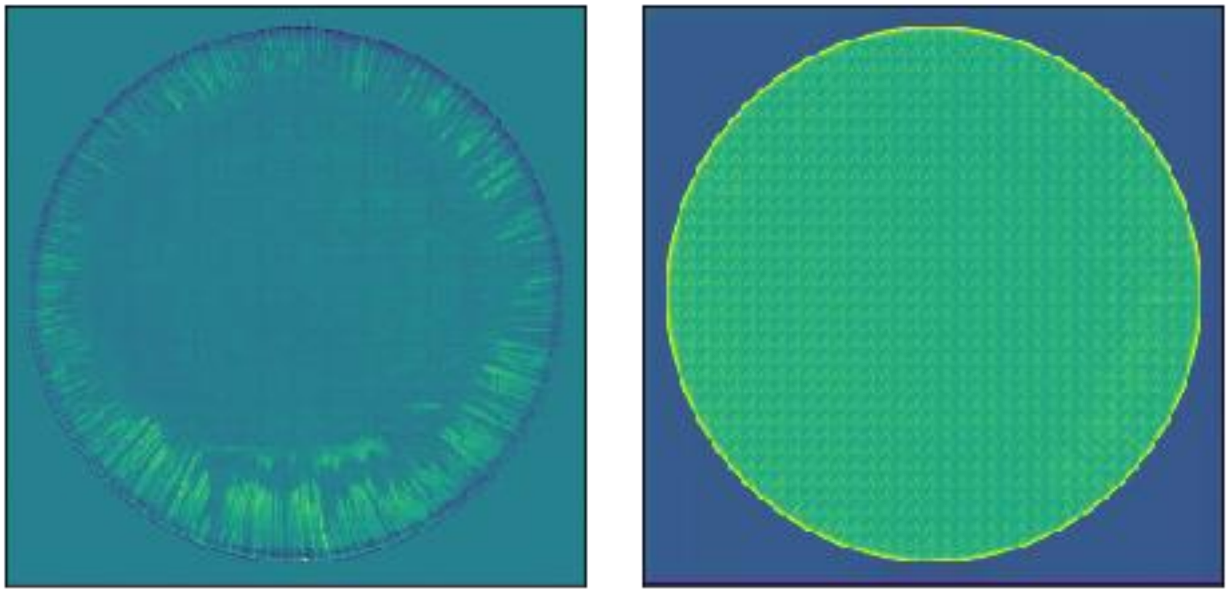

Figure 1. Corrected vs. uncorrected images for the same wafer, both gray-scaled.

Many different classifiers were trained and tested using these corrected images: neural network, logistic regression, RBF SVM, random forest, and KNN. One aspect of this dataset which proved to be a challenge for training the classifiers was the large disparity in class frequency between the good and bad wafers. Since there were only 16 bad wafers compared to over 8000 good wafers, the models could achieve over 99% accuracy by classifying all of the wafers as good. Despite this seemingly good performance, obviously this isn't helpful since it's just revealing something that's already known (that most of the wafers are good) with no progress towards the actual goal (finding which wafers are bad). In terms of classification metrics, an ideal solution for this problem would have very high recall (i.e. few false negatives) and moderate precision (i.e. some false positives). False negatives are much more costly than false positives in this context since FNs would mean true bad wafers escaping down the line and not being detected until later (the worst case being that they aren't detected until after having been shipped to the customers), whereas FPs can be identified easily by simply manually reviewing the images that were classified as bad. The cost of FPs would be in the man hours required to complete the manual image review, which is insignificant by comparison if the number of wafers classified as bad is small relative to the size of the dataset as a whole.

The first tests that were completed with the neural network model, training and testing on the full dataset, tended to produce this type of result: all wafers classified as good in most trials, and occasionally all wafers classified as bad depending on the parameter combination that was used (also equally unhelpful, despite the 100% recall).

	Predicted: 0 (Good)	Predicted: 1 (Bad)
Actual: 0 (Good)	8730	0
Actual: 1 (Bad)	16	0

Table 2. Confusion matrix for MLP classifier (alpha = 1000, hidden layer sizes = (40,)), trained and tested on full dataset (corrected images).

The strategy used to combat this was to take a small subset of the dataset to use for training such that the ratio of bad wafers to good wouldn't be so miniscule as in the full set. All 16 bad wafers were taken in the training set every time, and the remaining data points were randomly sampled from the good wafers in the dataset. The maximum size of the training set where the model was consistently able to correctly classify all of the bad wafers seemed to be around 300 total data points. The KNN and RBF SVM models trained with this set of 300 points were able to achieve 100% classification accuracy when testing on both the training set and the full dataset. This behavior was also consistent for training sets of different sizes (100 points, 200 points, etc.) up to 300. Other classifiers that were tested also were able to achieve 100% recall by training on a subset of the data (which included all of the bad wafers) and testing on the full dataset.

Model	Recall	Precision	Sample Size
Neural network (alpha=1000, hls=(40,))	0.4375	0.25	100
KNN (k=1)	1	1	300
Logistic Regression (C=10)	1	0.76	300
Random Forest (n_trees=1000)	1	0.57	500
RBF SVM (C=100, gamma=100)	1	1	500

Table 3. Classification results from different models trained on a subset of the data and tested on the full dataset.

This result was encouraging as it proved that some of the classifiers were able to distinguish bad wafers from good. However, this test doesn't prove that these models would be able to find bad wafers in an unseen dataset that the models weren't trained on. To assess this capability of these models using the data available, the bad wafers were split between training and test sets and the parameters of the various classifiers were tuned to achieve the best possible performance. Nearly all of the models that were tested across different types of classifiers with different parameter tunings were not able to pick out the bad wafers from the test set that the model wasn't trained on. The most common outcome on the test set was for the model to mark 100% of the wafers as good, which happened consistently regardless of the size of the dataset used, the relative proportions of data points assigned to the training and test splits, and whether the data was stratified by label and/or shuffled. Some of the models that were tested did correctly classify a portion of the bad wafers, but those models often also had a large amount of false positives, usually larger than the amount of true positives. While having some false positives is acceptable in this problem for the reasons discussed previously, a higher rate of false positives than true positives in the test set implies that the false positives when applying the model to a full dataset would number in the thousands. This means that thousands of wafers would require manual review, which effectively defeats the purpose of the classification in the first place.

Model	Recall	Precision	Total Sample Size
Neural Network (alpha=1000, hls=(40,))	0	0	300
KNN (k=1)	0.17	0	300
Logistic Regression (C=100)	0	0	300
Random Forest (n_trees = 100)	0	0	300
RBF SVM (C=100, gamma=100)	0	0	300

Table 4. Example classification results for different models trained/tested using a subset of data split 0.67/0.33.

Another training method that was tested involved training the models with a small subset of the full dataset which did not include all of the bad wafers, and then testing those models on the full dataset to see how many of the bad wafers were correctly classified. In most cases, the number of true positives did not exceed the number of bad wafers included in the training set. For example, if the training set contained 12 out of the 16 bad wafers, the models trained using that set would only produce up to 12 true positives, those being the wafers that were in the training set. The only exceptions to this would be cases where the model classified all or a majority of the wafers in the full set as bad, but again, this type of model isn't useful for the goal at hand.

Model	**Recall**	**Precision**	**Training Size**
Neural Network (alpha=1000, hls=(40,))	0	0	200
KNN (k=1)	0.75	0.92	200
Logistic Regression (C=100)	0.81	0.35	200
Random Forest (n_trees = 100)	0.63	0.43	200
RBF SVM (C=100, gamma=100)	0.50	1	200

Table 5. Example classification results for different models trained with a subset of data not including all bad wafers and tested on the full dataset.

One question that came to mind along the lines of the test described above is whether the particular bad wafers included in the training set might have an impact on the resulting classification accuracy. This is relevant to consider since in addition to the fact that the total number of bad wafers in the dataset is small, the size and contrast of the signature varies significantly between the bad wafers. Intuitively, the bad wafers on which the signature covers the smallest area of the wafers and/or has the lowest contrast on the corrected images would be the most likely to be misclassified if they were included only in the test set but not in the training set. Conversely, if these more subtle wafers were included in the training set and then the test set included wafers with the same "style" of signature but covering a larger area of the wafer and/or with higher contrast, those wafers should be much more likely to be classified correctly since the wafers in the test set show the same type of signature as the wafers in the training set but with "higher intensity". See Figure 2 below for an example of one of the subtle bad wafers compared to another wafer with a more obvious signature. The bad wafers were manually allocated to training or test sets accordingly (11 in the training set, 5 in the test set), and the remaining data points in each set were randomly sampled from the good wafers. This training strategy was tested with several different sample sizes, different types of classification models, applying the trained model to the test set or the full set, but the results still largely came out the same as before. In most of the trials that were completed, 100% of the data points in the test set were classified as good, whereas most of the trials that had more than 0 true positives had most of the data points classified as bad. The models trained using this strategy did have better success when testing on the full dataset, but again, the correctly classified bad wafers always ended up being those that the model was trained on.

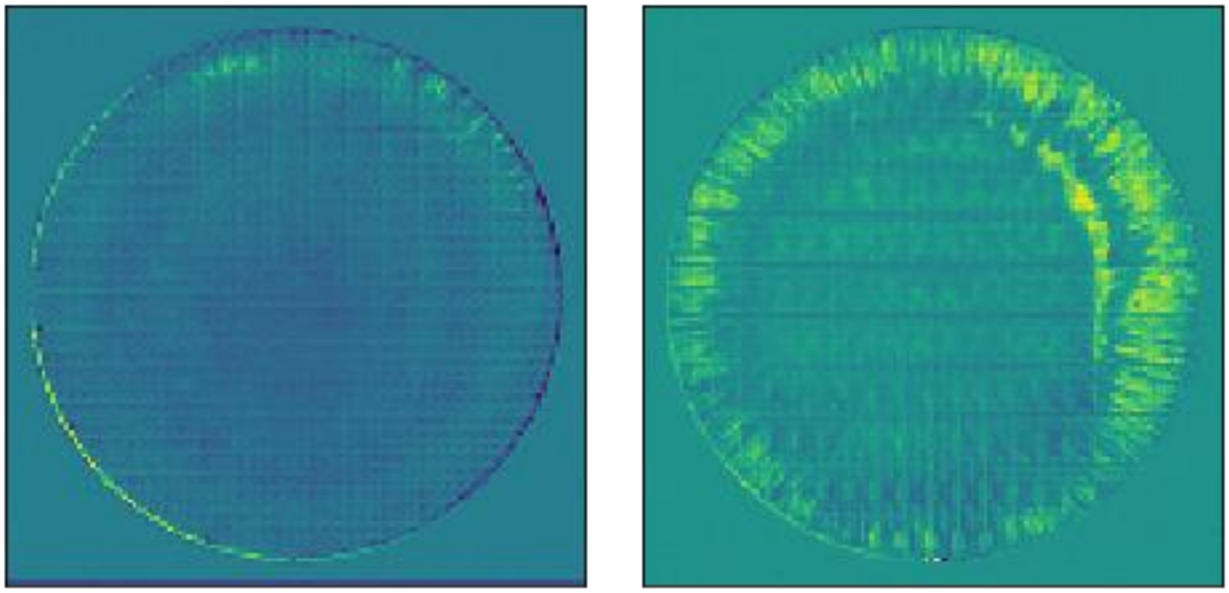
Figure 2. Comparison between a wafer with a subtle signature (left) vs. a more obvious signature (right).

Model	Recall	Precision	Total Sample Size
Neural Network (alpha=1000, hls=(40,))	0	0	300
KNN (k=1)	0	0	300
Logistic Regression (C=100)	0	0	300
Random Forest (n_trees = 100)	0	0	300
RBF SVM (C=100, gamma=100)	0	0	300

Table 5. Example classification results for different models trained using 11 of 16 manually selected bad wafers and tested using the other 5.

The next idea was to try a slight adjustment to the pre-processing method. Up to this point, all of the trials had been done with the images after block-reducing 8x in each dimension, gray-scaling, subtracting the lot reference image, and reducing the dimension further using PCA. This degree of block reduction was selected as the minimum required for all the images to be stored in memory. However, a higher degree of block reduction intuitively could be beneficial for this problem for a few reasons: 1) Block reduction naturally smooths out the noise of a given image, so perhaps increasing the degree would help to get rid of some of the non-significant variation between images driven by differences between product groups, and 2) even if the classification accuracy didn't improve, the reduction in data size would help to speed up the computations. In this test, the degree of block reduction was increased from 8x to 16x in each dimension, and the number of principal components retained after PCA was also reduced proportionally from 80 to 40. The same classifiers were tested on this dataset with the same training methods (training on a small subset of the data and testing on the full dataset, splitting the bad wafers between training and test sets, etc.) and the model parameters were tuned to achieve as close to the desired performance (high recall, moderate precision) as possible. Unfortunately, the results of this test ended up being very similar to the previous tests: models were usually only able to correctly classify bad wafers that they had been trained on, other than cases where the model classified most of the wafers as bad.

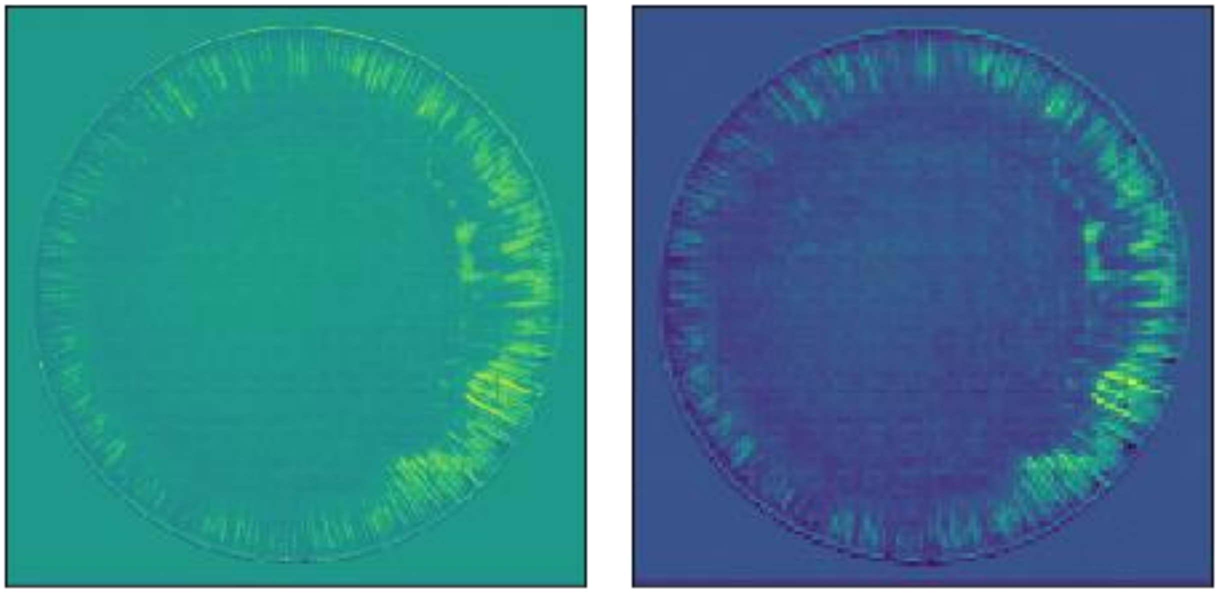
Figure 3. Comparison of corrected images for the same wafer with 8x (left) vs. 16x (right) block reduction

The final training method that was tested, and the one that produced the best results, was using different weights for the two classes with models that have this as an input parameter (logistic regression, random forest, SVM). Since the relative frequencies of the two classes in this dataset and the relative costs of false negatives vs. false positives are both so mismatched, it makes intuitive sense that weighting the two classes equally in training (as the models do by default) may not be the best strategy for finding the most effective classification model. The weight on class 0 (good wafers) was kept at 1 while the weight on class 1 (bad wafers) was increased by factors of 10 with each classifier until the best combination of parameters was found to achieve optimal classification performance. See the table below for the best results that were achieved with each classifier. Each model was trained with the set including only the 11 manually selected bad wafers and tested on the full dataset. Out of the classifiers that were tested, logistic regression tended to give the best results, both in terms of the best recall/precision values and also in terms of its consistency/robustness across different C values and weights. Although none of these models were able to achieve 100% recall without also having a large number of false positives, logistic regression was able to outperform the training set by correctly classifying 12/16 bad wafers (with only 11 of those being in the training set) under multiple different combinations of C and weight values.

Model	**Recall**	**Precision**	**Weights (1/0)**
Logistic Regression (C = 372)	0.75	0.19	1000/1
Random Forest (n_trees = 100)	0.6875	1	1000/1
Linear SVM (C = 0.05)	0.75	0.18	1E10/1

Table 6. Best classification results achieved by tuning class weights.

Evaluation and Final Results:

The goal of this project was to find a pre-processing/classification strategy that would successfully be able to classify unseen wafers as having a given signature after being trained on a set of known bad wafers with similar signatures. Some of the factors that make this a challenging problem are that the data are high-dimensional, there isn't a way to collect more examples of bad wafers beyond the ones that are already available, and while to the human eye it's easy to tell that all of the signatures on the bad wafers are caused by the same thing, finding a model that can generalize from known bad wafers to unseen bad wafers is not trivial. Since the patterns that the classifiers are intended to capture are highly non-linear and show a good amount of variability in terms of the specific shape of the signature, the region of the wafer that it's in, and the rotational orientation, intuition would predict that a neural network classifier (or something similar) would be the best-suited for this task. However, in practice, the neural network models failed to produce anywhere near the best results for this scenario, even in cases where other models performed quite well such as training and testing on a small subset of the full dataset where all of the bad wafers showed up in both sets. While the only type of neural network tested here was a multi-layer perceptron classifier, it is possible that a convolutional neural network would be more capable of producing better results because of its ability to extract hierarchical feature representations from sets of images. The small sample size could still make this a challenging task, but this would be an interesting avenue to explore for future testing.

The classifier that tended to produce the best results with the most robustness across different testing conditions was logistic regression. This was another surprising result given that LR is a linear classifier, but it seems that the linear vs. non-linear classifier distinction may not be very useful for predicting which model performs the best on this data (at least not in the sense that the former should be worse and the latter better). However, despite showing the best performance, even LR was not able to correctly classify impacted wafers that the model wasn't trained on as bad with any kind of consistency. This is ultimately the most important factor for determining whether any of these models would be useful in a real situation, so even the best logistic regression model that was found here would not yet make the cut to be deployed in production.

Aside from the classifiers themselves, there could also be more interesting possibilities to explore with the data pre-processing. The higher degree of block reduction (16x vs. 8x) didn't seem to have a negative impact on the classification; it seemed to even increase the contrast of the defect signature relative to the rest of the image (in addition to speeding up the computation time), so this will be a keeper moving forward. Another aspect that the pre-processing could address is the other sources of variation in the images outside of the defect signatures themselves. One way to do this might be by removing the wafer edge and everything outside of it from the images since the wafer edge is a major source of variability in these images that is not at all relevant for the task at hand. This could be done by applying a circular mask with a radius just smaller than that of the wafer and setting all pixel values outside of that to null, although it's not clear how PCA would work with the images after this treatment. Another possibility would be to take a subset of each image and use that for the classification instead of the full images themselves. For example, one notable feature of the corrected images is that the defect signatures tend to show up clearly to the human eye with high contrast. This high contrast indicates that there are small regions within those images where the pixel values have a high rate of change (in fact, this is true for most events where WIS bounding is used, not just this event). Therefore, maybe if one row/column that intersects the signature was taken from the image and used to train the classifiers, that could produce better results than using the full images. Furthermore, if the resulting model proved to be robust across different types of signatures, even those which haven’t been observed yet, that would have the potential to be an incredibly powerful tool for defect detection in semi-conductor manufacturing. These ideas, among others, will all be worth exploring in the future.