Detecting Phases in Steel Microstructure

Suraj Singh
[Roll No. : CS-1718]

Master's Thesis

In partial fulfillment of the requirements for the degree of

Master of Technology in Computer Science

Supervised by

Dr. Dipti Prasad Mukherjee

submitted to

Indian Statistical Institute

Kolkata, July 2019

This document is set in Palatino, compiled with pdfLaTeX2e and Biber.

The LaTeX template from Karl Voit is based on KOMA script and can be found online: https://github.com/novoid/LaTeX-KOMA-template

Certificate

This is to certify that the dissertation titled **"Detecting Phases in Steel Microstructure'** submitted by **Suraj Singh** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a *bona fide* record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Dipti Prasad Mukherjee
Professor & Head,
Electronics and Communication Sciences Unit,
Indian Statistical Institute,
Kolkata-700108, India.

Acknowledgment

I would like to express my sincere gratitude towards my supervisor Prof. Dipti Prasad Mukherjee for constantly guiding me in my work. He has always been by my side in times of doubt and has provided necessary hints when I got stuck.
My sincere thanks to Yusuf Akhtar for his valuable suggestions and discussions. I would also like to thank my friends, who are keen to listen to my doubts.

Abstract

Detecting Phases in Steel Microstructure is one of the interesting problem in the field of computer vision. In this work, we discuss a pixel based classification approach. A classifier is only as good as the information you give it. On the other hand it may have a huge intrinsic disproportion the number of examples in each class, Which hinder the classification performance. There are many ways you can adjust how you're representing your input data for learning of model. In this paper, we propose ensemble method based on outlier detection to comprehend better the data used as a part of learning of model , which is random forest and discuss it merits and demerits of other related method which we use.

.

Contents

List of Figures

1 Introduction

1.1 Problem Description

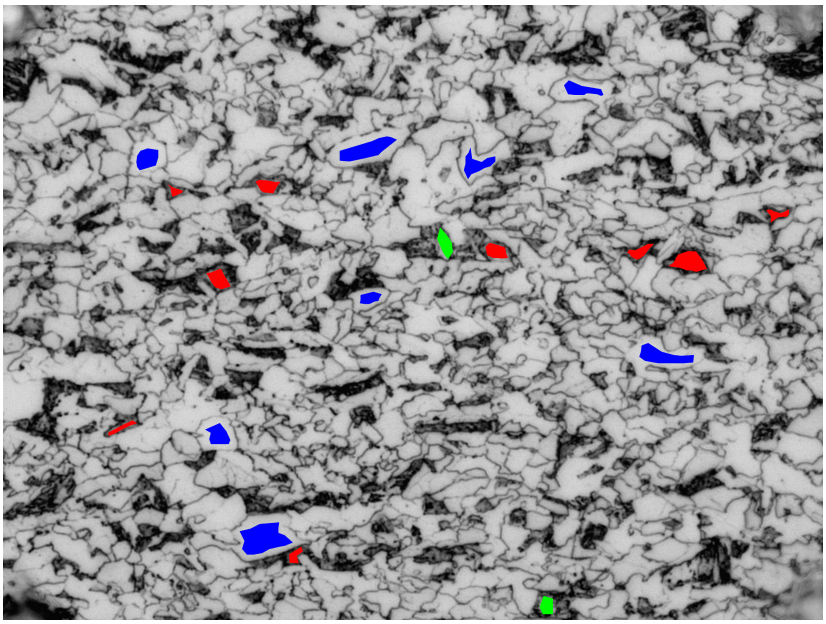
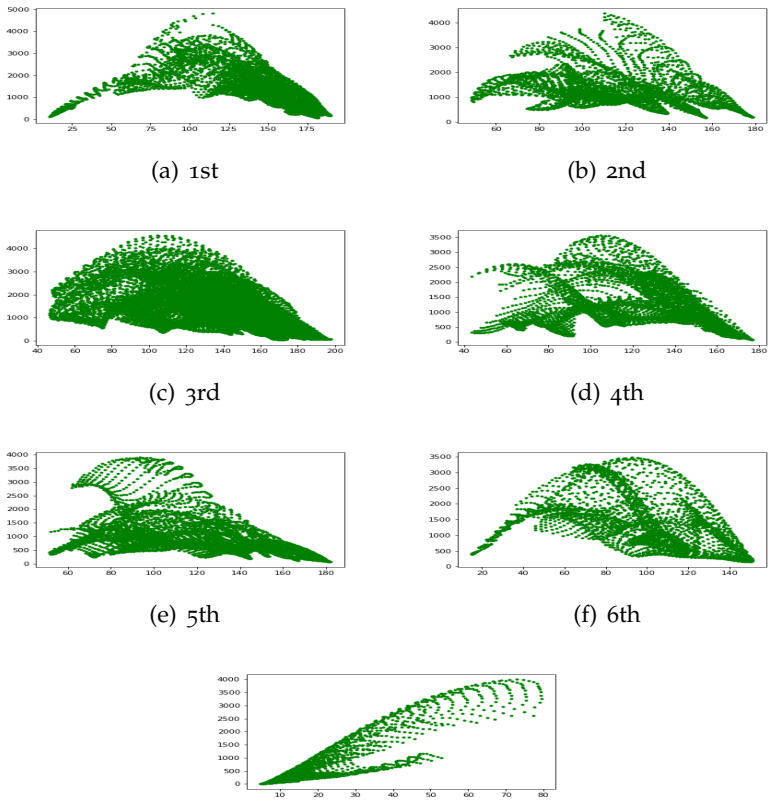


Figure 1.1: Ground truth image

We have twenty ground truth of steel microscopic images one of them shown in figure 1.1, containing some portions of themselves labelled in blue, green and red respectively. The region marked by red, green and blue in the ground truth image are represent class label correspond to the phase martensite, bainite and ferrite respectively. Our goal is to detect these phases in steel Microstructure.

1.2 Challenges



(a) 1st (b) 2nd

(c) 3rd (d) 4th

(e) 5th (f) 6th

(g) 7th

Figure 1.2: Distribution of feature space from seven different images correspond to same class label which is bainite.

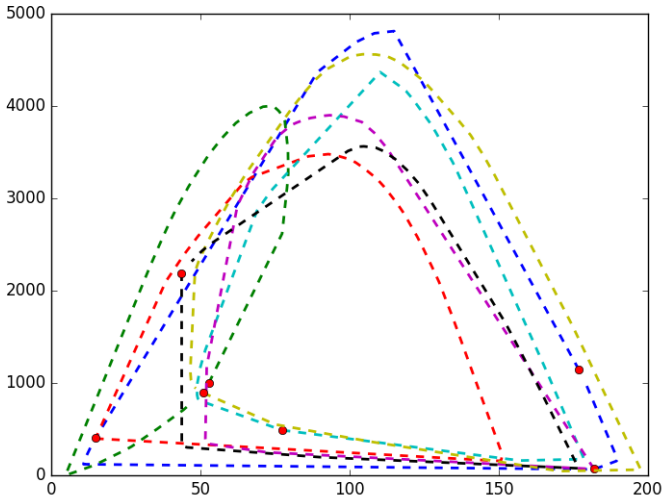


Figure 1.3: Distributions of feature space from seven different images correspond to same class label which is bainite in single frame

In figure1.3 we are trying to reflect relationship between seven feature spaces which is shown in figure1.2 by seven different color of convex hull boundaries. Blue, cyan, yellow, black, magenta, red, and green color are used for convex hull boundary correspond to figure1.2(a),figure1.2(b), figure1.2(c), figure1.2(d), figure1.2(e), figure1.2(f), figure1.2(g) respectively. The feature that we are used in figure1.2 discussed next.
If (x,y) is a pixel location which is determine by ground truth image, we are taking a $33 * 33$ sliding window around (x,y) which is defined by the spatial position {(x+i,y+j): i ,j $\in$ [-16,16] }. Then we calculate pixel based feature mean and variance in this sliding window as our feature vector. In this way, by taking mean along horizontal axis and variance along vertical axis respectively, we get our desire feature.
The reason behind this discussion is, we assume that distribution of feature space correspond to same class label from different images are approximately same. But here distributions are not.

1.3 Objective

Our objective is to detect phases in steel Microstructure. For this purpose, we used pixel based classification approach but performance of classifier depend on nature of feature space . In our case distribution of feature space is not approximately same, which hinder the performance of classifier. We assume that it may cause of due to operator variance . Our objective is balancing distribution of feature space such that it approximately same for same class label from different images.

1.4 Our contribution

In this work, we proposed a construction on outlier detection to comprehend better the data used as a part of learning of model, which is motivated by ensemble based method[Wu and Nagahashi, 2015Mishina et al., 2015]. In our problem we have total three classes. Goal of our proposed construction is considering feature space of each classes individually and find better distribution on it by removing outliers.

2 Preliminaries

2.1 Overview of Isolation Forest for outlier detection

2.1.1 Brief overview of outlier

An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism from other observations.
The following question then becomes important: Why outlier are important? The ability of a classifier to recognize unknown inputs is important for many classification-based systems. In training, a machine learns from ordinary data. Later, using previously unknown data, this machine tries to separate ordinary from novel patterns. Where the goal is to recognize whether an input is from the known set of classes and from which specific class, or from an unknown domain and does not belong to any of the known classes.

2.1.2 Basic idea of Isolation Forest

Isolation Forest [Liu, Ting, and Zhou, 2012] is a popular outlier detection algorithm. The term isolation means 'separating an instance from the rest of the instances'. If we try to segregate a point which is obviously a non-outlier, it'll have many points in its round, so that it will be really difficult to isolate. On the other hand, if the point is an outlier, it'll be alone and we'll find it very easily. Since outliers are few and different and therefore they are more susceptible to isolation.

Figure 2.1: Isolation of an inlier

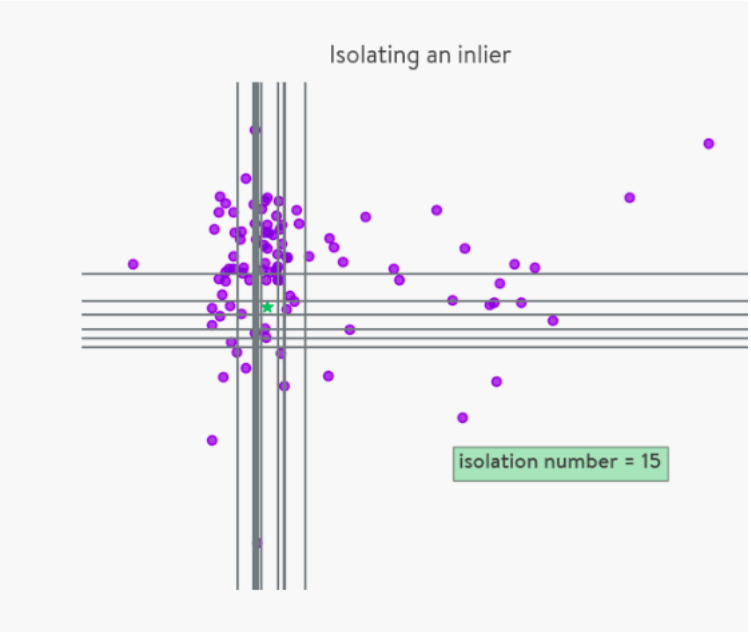
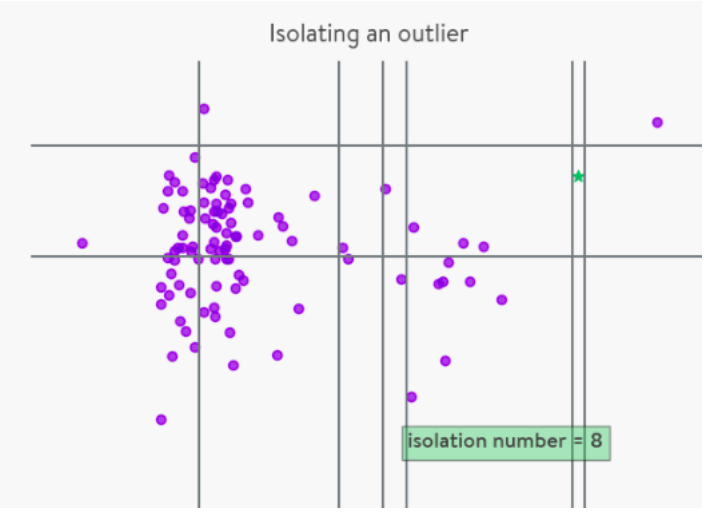


Figure 2.2: Isolation of an outlier



In a data-induced random tree (known as Isolation tree), partitioning of instances are repeated recursively until all instances are isolated. This random partitioning produces noticeable shorter paths for outlier since
(a)the fewer instances of outliers result in a smaller number of partitions – shorter paths in a tree structure, and
(b)instances with distinguishable attribute-values are more likely to be separated in early partitioning. Hence, when a forest of random trees collectively produce shorter path lengths for some particular points, then they are highly likely to be outliers.

2.1.3 Isolation tree working procedure

For each tree:
1. Get a sample of the data
2. Randomly select a dimension or feature say x.
3. Randomly pick a value in that dimension or feature(which is chosen in previous step) say x1.
4. Draw a straight line through the data at that value x1 and split data in two parts. One part having value less or equal to x1 for x and another part having value greater than x1.
5. Repeat until tree is complete in other word repeat the above process until u can split(if all data points in a splitting have only one value to corresponding randomly chosen dimension u can't split).

- Anomalies will be isolated in only a few steps.
- Nominal points in more.

2.1.4 How to measure outlier from isolation forest?

We employ path length as a measure of the degree of susceptibility to isolation,which is known as isolation number. A point x is outlier or inlier is determined by the Path Length say h(x) for x, which is measured by the number of edges x traverses an iTree from the root node until the traversal is terminated at leaf node.

- Short path length means high susceptibility to isolation.
- Long path length means low susceptibility to isolation.

Since building an isolation tree depend on random choice of attribute and random value of corresponding attribute ,to get better result we repeatedly create isolation tree and combination of these isolation tree create isolation forest .
The final isolation number of a point say x is a combination of all isolation numbers for each isolation tree. In this way we have isolation number for each data point we choose randomly a threshold between minimum and maximum of set of all isolation numbers and plot the points whose isolation number is less than threshold as outlier. Now since threshold is chosen randomly it is may not be desirable threshold but we know the fact that less isolation number implies outlier, so we update our threshold accordingly,by continue this process we get our desire outliers.

2.2 Brief overview of random forests

We assume that the user knows about the construction of single classification trees.' Random Forests grows many classification trees. To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest)[Liaw, Wiener, et al., 2002Breiman, 2001].

2.2.1 Training procedure for random forest

1. Randomly select "k" features from total "m" features, where k $\ll$ m.
2. Among the "k" features, calculate the node "d" using the best split point.
3. Split the node into daughter nodes using the best split.
4. Repeat 1 to 3 steps.

Build forest by repeating steps 1 to 4 for "n" number times to create "n" number of trees.

2.2.2 Random forest prediction

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome.
2. Calculate the votes for each predicted target.
3. Consider the high voted predicted target as the final prediction from the random forest algorithm.

3 Proposed Method

3.1 Introduction

Assume that we have N number of images containing some portions of themselves labelled in blue, green and red respectively. Let the i^{th} image be referred by I_i. Let the set of all feature vectors by pixels labelled as red in the I_i be referred by R_i. Similarly, let the set of all feature vectors of pixels labelled as green and blue in I_i, be referred by G_i and B_i respectively.

3.2 Objective

We assume that the class label of a test feature vector F_t is identical to the class label of the nearest feature vector(based on euclidean distance). It is apparent that the goal of the proposal is to predict the chance of a feature vector belonging to class say red: 1 for inlier, 0 for outlier. Let us have a method, say M that assign outlier/inlier labels to the feature vector in R_i.

3.3 Motivation

There are now two possible approaches to find the label of a test feature vector F_t. Assume that the distribution of the feature vectors in the feature space for every R_i is approximately the same. There are two possible scenario.

- 1. Take the union of all R_i. Find the label of F_t based on the nearest distance from the feature vectors in the union of R_i.or

- 2. Take any R_i. Find the label of F_t based on the nearest distance from the feature vectors in R_i.

Both the above methods would return the same result if the distribution in the feature space of the feature vectors R_i in every image I_i is approximately identical. However it is reasonable to assume that these distributions of the feature vectors in the feature space need not be identical. The cause may be operator variance. The goal of the proposal is to correct this operator variance that has inevitably crept into the ground truth. If indeed the operator variance has crept into the ground truth, there would be discrepancies in point (2) since choosing a different R_i may return a different class label to F_t. Considering all the class labels assign to F_t we may come to a single decision regarding the class label of F_t with the aid of ensemble classifier [Wu and Nagahashi, 2015Markou and Singh, 2003 Mishina et al., 2015]. This motivates us to choose point(2) that is replacing any R_i by all R_i one at a time for aiding us in labelling a test feature vector. This ensemble based classification approach is discussed next.

3.4 Methodology

Consider the image I_j. We determine class labels (inlier/outlier) of the feature vectors in R_j from our outlier detection algorithm Isolation Forest. From this we want to determine class labels (inlier/outlier) of test feature vector $F_t \in \cup R_i \setminus R_j$. However this approach may have high complexity if the size of R_i are large (since our approach based on euclidean distance of all test feature vectors from $\cup R_i \setminus R_j$ to R_j). So we come to a solution by taking random subset of R_i for representing possible values of class labels(inlier/outlier) of F_t.
In this context consider a subset of R_j referred by RS_j. Since class label of the feature vectors in R_j is known, it implies class labels(inlier/outlier) of RS_j is also known. Let V_R denote the union of RS_i i.e. $V_R = \cup RS_i$. The feature vectors in the set $V_R \setminus RS_j$ do not have class label assign to them for I_j.

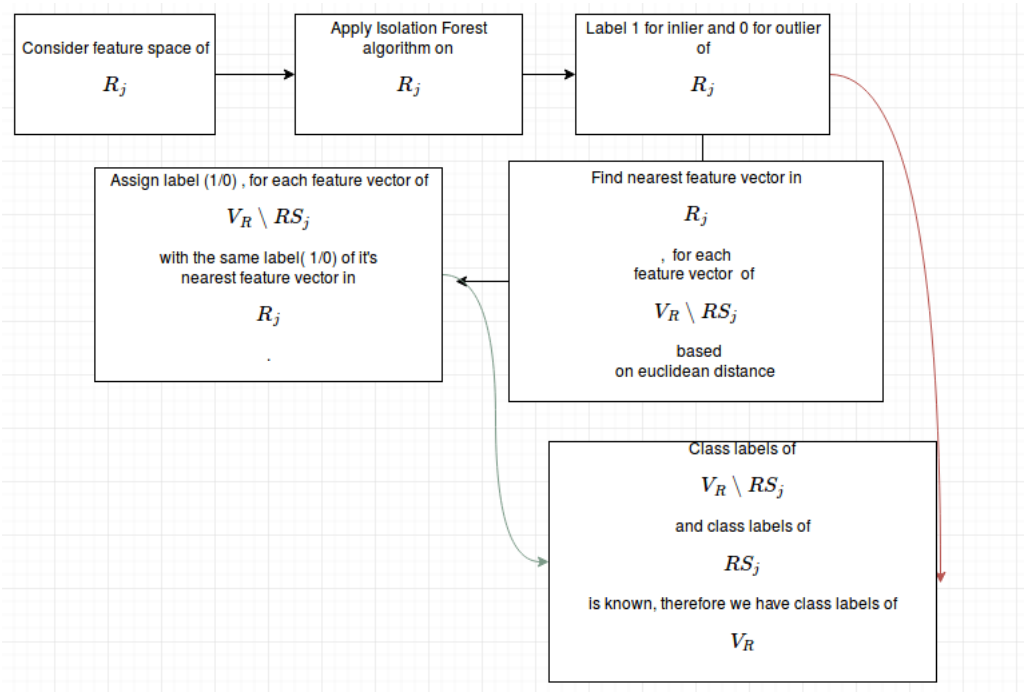


Figure 3.1: How to label test feature vector correspond to R_j

To assign class labels to these feature vectors in $V_R \setminus RS_j$, we can adopt the following algorithm whose block diagram shown in figure3.1.

Algorithm
1.Input: V_R, RS_j, R_j
2.Procedure:
(a) Find the euclidean distances of $F_t \in V_R \setminus SR_j$ from $\forall$ feature vector of R_j.
(b) Find the nearest feature vector of F_t say F_1 *in* R_j.
(c) Label of F_t = Label of F_1.
3.Output: Lable (inlier/outlier) of feature vectors in V_R.

In the proceeding paragraph we stated that we have the class labels (inlier/outlier) for RS_j. From above algorithm, we have the class labels for

$V_R \setminus RS_j$. Therefore we have the class labels for the set $RS_j \cup V_R \setminus RS_j$, which is V_R . We repeat this assignment of inlier/outlier class labels in V_R for every image I_j. Let the set of class labels assigned to the feature vector in V_R for image I_j be denoted by P_j.
To encapsulate the class labels assigned to the feature vectors in V_R for image I_j, we construct a single matrix of dimensions N*length(V_R) (say K=length(V_R). Let this matrix be denoted by C. The element C(a,b) refers to the class label of the b^{th} element in V_R for the a^{th} image I_a. In other words the a^{th} row in C is P_a. Now we consider a column (say the b^{th} column) of C. This column contains all the class labels assigned to the b^{th} feature vector in V_R for all images I_j. Since these class labels may not be consistent with one another the question arises that how should we come to a single class label regarding the b^{th} feature vector,say F_b.
Assume that we have chosen a class label for F_b arbitrarily. This is repeated for the remaining feature vectors in V_R . Let the set of these new class label referred by P_s. The question arises that whether there is a better choice of P_a. We may assume that P_a generated by method M, eliminates most, if not completely the effect of operator variance by assigning appropriate class labels (inlier/outlier) to the feature vector correspond to R_a. If we superimpose P_s on P_a, it may so happen that feature vectors correspond to R_a which are outlier treated as inliers and vice verse. Therefore we may assume that more the difference between P_s and P_a the more the operator variance is reflected in the labels assigned to the feature vectors in R_a . Let m_a denote the mismatch between P_a and P_s. This is rewritten below.
$m_a = \text{count}(P_s(k) \neq P_a(k)), \forall$ indices $k \in V_R$.
A chosen P_s may perform differently for a different choice of 'a' based on the value of m_a. So for all m_a values to reflect how good is the chosen P_s our objective function is $m(P_s) = \sum_{i=1}^{N} m_a$. Therefore the ideal solution is, say P^{ideal} for which our objective function attain its minimum m(P^{ideal}).

3.5 P^{ideal} estimation

Let $P^{majority}$ be our set of all possible class label for V_R, which we obtain by majority voting. Therefore len(V_R) = len($P^{majority}$) and b^{th} element of $P^{majority}$ represent the class label of b^{th} feature vector in V_R .

In more explicitly, b^{th} element of $P^{majority}$ is majority element in b^{th} column of all P_a.
claim: P^{ideal}=$P^{majority}$.
For $P^{majority}$, in a column number of mismatch is less than $N/2$ that is why column of $P^{majority}$ is chosen on majority basis. Now suppose P^{ideal} differ in x number of column position from $P^{majority}$.
Then number of mismatch correspond to $P^{majority}$ in this x column is less than or equal to $x * N/2$ but number of mismatch correspond to P^{ideal} must be greater than or equal to x*N/2 that is why x number of column of P^{ideal} is not majority element.
Therefore , m(P^{ideal}) - m($P^{majority}$) ≥ 0. Therefore, $P^{majority}$ is the our optimal solution or, P^{ideal} = $P^{majority}$.

4 Experiments and results

In this section, we compare two method say $M1$ and $M2$ (based on our proposed method. Goal of method $M1$ is choosing proper feature to achieve better accuracy, whereas goal of method $M2$ is controlling operator variance by our proposed method which is discuss in previous chapter 3.4 for further improvement of method $M1$ by choosing proper feature which is same as feature used in method $M1$.

4.1 Training phase of method $M1$

4.1.1 Feature extraction of method $M1$

If (x,y) is a pixel location which is determine by ground truth image, we are taking a $33 * 33$ sliding window around (x,y) which is defined by the spatial position {(x+i,y+j): i ,j $\in$ [-16,16] }. Now we divide this sliding window into small block with size $3 * 3$. Since sliding window size is $33 * 33$ and each block size is $3 * 3$, we have $11 * 11$ number of blocks. Calculate pixel based feature mean and variance for each block.
Next, by taking X axis as mean and Y axis as variance, we create a $3d$ histogram with 10 number of bins for X axis and 10 number of bin for Y axis that is $10 * 10$ number of cells.

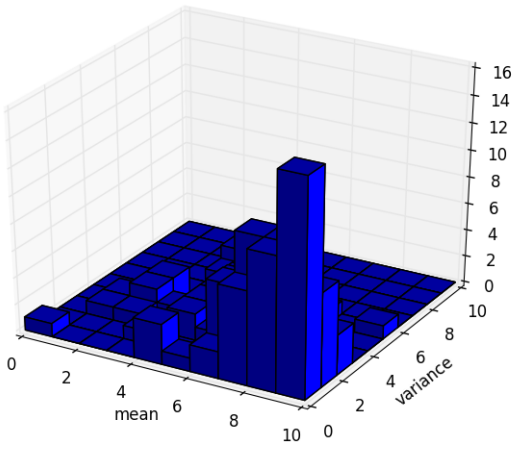


Figure 4.1: 3*D* histogram

Where bin length for X and Y is defined as $l1$=maximum(X)−minimum(X)/10 and $l2$=maximum(Y)−minimum(Y)/10 respectively. The number of occurrences of each x and y for X and Y into $10 * 10$ cells give us 100 count. By concatenating each row we obtain desire feature of dimension 100[Paul et al., 2018].

4.1.2 Learning phase of method $M1$

For training purpose , We chose 16 images among 20 images based on 5-fold cross-validation technique . Then from the chosen feature we train our classification model random forest with 150 number of trees.

4.2 Training phase for method M2

4.2.1 Feature extraction for method $M2$

If (x,y) is a pixel location which is determine by ground truth image, we are taking a 33*33 sliding window around (x,y) which is defined by the spatial position {(x+i,y+j): i and j takes integer value in [-16,16]}. Then we calculate pixel based feature mean and variance in this sliding window as our 2 dimensional feature vector.

4.2.2 Learning phase of method $M2$

For training purpose, We chose 16 images among 20 images based on 5-fold cross-validation technique.
Let say feature space correspond to the region marked by red, green and blue in 16 images referred by T_R, T_G and T_B respectively.
In the next step to get better distribution our goal is to apply ensemble based outlier detection technique which is based on our proposed construction 3.4. But as our size of data set for training is too large, We choose a subset of feature from T_R which referred by ST_R. Thus we divide our feature space T_R correspond to the region marked by red into two parts. One part is ST_R and other part is $T_R \setminus ST_R$. Let $T_R \setminus ST_R$ be referred by Tr_R. In the next step, We apply our ensemble based outlier detection technique which is based on our proposed construction in 3.4 on ST_R. Let SI_R be inlier in ST_R that we got in the previous step. Finally we Keep $Tr_R \cup SI_R$ for training purpose and transform the feature space of $Tr_R \cup SI_R$ into 100 dimensional feature space which is same as used in method $M1$.

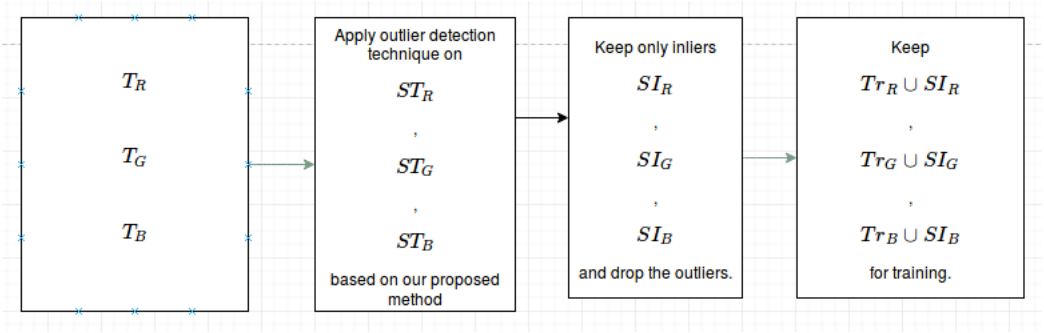


Figure 4.2: Steps in training phase

Similarly we choose subset of feature from T_G and T_B which are referred by ST_G and ST_B respectively. Let $T_G \setminus ST_G$ and $T_B \setminus ST_B$ are referred by Tr_G and Tr_B respectively. Then we apply ensemble based outlier detection technique which is based on our proposed construction 3.4 on ST_G and ST_B respectively. Let SI_G and SI_B inlier in ST_G and ST_B respectively. We keep $Tr_G \cup SI_G$ and $Tr_B \cup SI_B$ for training purpose and transform the feature spaces into 100 dimensional feature space which same as used in method $M1$.
By this 100 dimensional feature space correspond to $Tr_R \cup SI_R$, $Tr_G \cup SI_G$ and $Tr_B \cup SI_B$, we train our classification model random forest with 150 number of tree, which is same as our classification model for method $M1$.

4.3 Testing phase of method $M1$ and $M2$

In testing phase of method $M1$ and method $M2$ we choose four images among twenty images by 5-fold cross validation technique. Our feature for testing phase for method $M1$ and $M2$ is same feature which is 100 dimensional feature (discussed in feature selection of method M1) . Note, we are not considering outlier detction technique for the testing phase of method $M2$.

4.3.1 classification Result

Let the portion in the data set marked by red, green and blue represented by class 1, class2 and class3 respectively.

Classification result for method $M1$			
Label	precision	recall	f1-score
Class 1	0.71	0.96	0.81
Class 2	0.83	0.20	0.32
Class 3	0.98	1.00	0.99
avg/total	0.88	0.86	0.83

Classification result for method $M2$			
Label	precision	recall	f1-score
Class 1	0.45894402	0.47396143	0.46633185
Class 2	0.62132643	0.40423428	0.48980316
Class 3	0.75273371	0.78549536	0.76876565
avg/total	0.61100138666	0.55455369	0.57496688666

Accuracy for method $M1$ = 0.8648409196270669
Accuracy for method $M2$ =0.6627247176114971

5 Discussion and Future Work

Though we are using outlier detection technique, since we are considering a sliding window of size $33 * 33$, so it is worthy to save the samples near the borderline and get rid by morphological or some other techniques. Whether the border line data is noise or not is important for forming the correct decision boundary.
Our proposed method involve nearest neighbour distance method for which the model has to run through the entire data set to compute distances and then find the nearest neighbors. Thus we require high time complexity and it is slow. For this reason we are considering a subset of feature from training not all the feature which may cause of insufficient information for learning.
Our interpolation technique based on ensemble method which depend on Performance of Isolation forest algorithm, so instead of nearest neighbour, we can do it by K nearest neighbour or some other technique.
The feature that we used is 100 dimensional but it may happen that all 100 features are not necessary, so for better performance we can adopt PCA or some other dimensional reduction technique.
In the future, various issues, such as since role of Isolation Forest algorithm have major impact in initial stage of our proposed method ,it should be considerable if there are any other technique perform better instead of Isolation forest.

Bibliography

Breiman, Leo (2001). "Random forests." In: *Machine learning* 45.1, pp. 5–32 (cit. on p. 9).

Liaw, Andy, Matthew Wiener, et al. (2002). "Classification and regression by randomForest." In: *R news* 2.3, pp. 18–22 (cit. on p. 9).

Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2012). "Isolation-based anomaly detection." In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6.1, p. 3 (cit. on p. 5).

Markou, Markos and Sameer Singh (2003). "Novelty detection: a review—part 1: statistical approaches." In: *Signal processing* 83.12, pp. 2481–2497 (cit. on p. 12).

Mishina, Yohei et al. (2015). "Boosted random forest." In: *IEICE Transactions on Information and systems* 98.9, pp. 1630–1636 (cit. on pp. 4, 12).

Paul, Angshuman et al. (2018). "Calculation of phase fraction in steel microstructure images using random forest classifier." In: *IET Image Processing* 12.8, pp. 1370–1377 (cit. on p. 17).

Wu, Shuqiong and Hiroshi Nagahashi (2015). "Analysis of generalization ability for different AdaBoost variants based on classification and regression trees." In: *Journal of Electrical and Computer Engineering* 2015, p. 8 (cit. on pp. 4, 12).